CONSTRUYA SU PROPIO SISTEMA EXPERTO

CHRIS NAYLOR

PROGRAMAS EN BASIC
PARA APPLE II Y SPECTRUM.
PROGRAMAS EN PASCAL.

DIAZ DE SANTOS, S. A.

Importador nº 60

MADRID

nc.

CF:

CT:

REF SP-4100/0048

SP-4100/0048.001

PTAS2.756

Chris Naylor

Construya su propio Sistema Experto

Traductor: Gregorio Fernández

Catedrático de Cibernética y Ordenadores
E.T.S.I. Telecomunicación
Universidad Politécnica de Madrid

DIAZ DE SANTOS, S.A.

MADRID / BARCELONA

Copyright © 1984 C. M. Naylor, 1983

Queda prohibida la reproducción total o parcial de la presente obra, bajo cualquiera de sus formas, gráfica o audiovisual, sin la autorización previa o escrita del editor, excepto citas en revistas, diarios o libros, siempre que se mencione la procedencia de las mismas.

Título original: «Build your own Expert System»

ISBN: 0-905104-412 (edición en lengua inglesa).
ISBN: 84-86251-33-8 (edición en lengua española).

Traductor: Gregorio Fernández

Diseño de portada: Francisco Bussalleu

Edita: DIAZ DE SANTOS, S. A.
C/ Juan Bravo, 3A
28006-MADRID

1.ª Edición: 1986.

Depósito Legal M-4286-1986.

Imprime: Imp. CALERO
Sancho Panza, 4 – 28018 MADRID

A Catherine y Philip
y que les aproveche…

Contenido

1. **El porqué de los sistemas expertos** 13

¿Para qué quiere usted un sistema experto?—¿Para qué quieren otras personas un sistema experto?—¿Qué es un sistema experto?—¿Qué quiere hacer usted con un sistema experto?—Sistemas expertos: algunas falsedades.

2. **Un esquema estadístico** 27

Elaboración de una matriz.—Probabilidades.—Más probabilidades.—Más variables.—El teorema de Bayes.

3. **Cómo eludir las probabilidades** 49

Cómo hacer que el ordenador se encargue del trabajo pesado.—El sistema de aprendizaje.—Otros tipos de datos. La regla de juicio.—Cómo construir una regla.—Probabilidades a priori.—Cómo tener más opciones.—¿Puede equivocarse el sistema?—Recapitulación: el programa actualizado.

4. **Cómo mejorar su experto** 83

Decisiones paralelas y secuenciales.—Cómo incluir algo de sentido común.—Una prueba para nuestro nuevo experto.

5. **Un experto en el mundo real** 111

De nuevo, el tiempo.—Un programa de ji cuadrado.—Cómo entrenar a su experto.—Cálculo directo.

6. **Funcionamiento del experto en casos reales** 133

Cómo utilizar su experto.—Juicio reservado.—El problema de la distancia.—La comprensión del problema.

7. Un experto en todas las cosas del mundo entero conocido **151**

Nodos.—Las variables, tal como están.—Cómo recorrer los nodos.—Nodos a la medida.—Instrucciones específicas.—Cómo salvaguardar su experto.—El código multinodo.—Algunos ejemplos.

8. Cómo utilizar su experto **199**

La elección de un problema.—El análisis del problema.

9. Sistemas expertos de gran envergadura **205**

MYCIN: Diagnósticos médicos.—PUFF: Trastornos respiratorios.—DENDRAL: Estructuras químicas.—PROSPECTOR: Búsqueda de minerales.—Otros ejemplos.
PECTOR: Búsqueda de minerales.—Otros ejemplos.

10. Un experto basado en reglas y escrito en BASIC **241**

Un sistema que funciona hacia atrás.—El programa BASIC.—Una base de conocimientos médicos.

11. La Torre de Babel **273**

12. Resumen y panorámica técnica 281

Sucesos.—Probabilidades.—Superficies.—Discriminación.—El algoritmo de aprendizaje.—Procedimientos paralelos y secuenciales.—Valores mínimos y máximos. Estrategia de procesamiento.—Conclusiones intermedias.—Interpolación lineal de respuestas.—Formatos de los datos.

13. Selección bibliográfica **297**

Indice de palabras **301**

Apéndice a la edición española **305**

Prefacio

En una librería, un lector o lectora en potencia tiene entre sus manos un ejemplar de un libro. Ya ha visto la portada y la contraportada, y ahora está leyendo el Prefacio. Según la Teoría, el Prefacio condiciona la venta.

Pero la Teoría puede decir lo que quiera. Si algún motivo puede inducirle a comprar este libro es que, para tratarse de un libro sobre informática, es relativamente barato. Además, incluye ejemplos que funcionan tanto para el Apple II como para el Sinclair Spectrum, de modo que, por el mismo dinero, se consiguen algunos programas gratis.

El libro le enseñará una serie de cosas sobre los sistemas expertos, pero, francamente, si yo le explicase aquí con todo detalle lo que va a aprender leyéndolo, podría inhibir el motivo de su compra. Al fin y al cabo, usted podría seguir ahí, en la librería, leer el Prefacio, y guardarse su dinero. Aunque sólo sea como aperitivo, le diré que podrá usted escribir su propio programa de diagnóstico médico, o un programa para averiguar por qué su coche no arranca por la mañana, o construir un sistema experto que aprende a base de ejemplos y que puede adquirir experiencia en áreas muy diversas.

También le enseñará bastantes cosas sobre estadística y sobre sistemas de inferencia, pero para todo eso tendrá usted que desembolsar lo que cuesta un ejemplar.

Ya sé que actualmente el dinero es escaso, y que podría usted gastárselo en alguna otra cosa más provechosa, como, por ejemplo, en unas copas, lo cual le proporcionaría mayor placer *a usted*, pero piense que yo también tengo que ganarme la vida, y que cuando conseguí terminar de escribir todo esto, solamente el coste de las cintas para la máquina de escribir ya me había supuesto un gasto importante.

Bien, después de todo lo anterior, a lo mejor lo he conmovido, ha desembolsado lo necesario para adquirir un ejemplar y ahora están pensando en leerlo. Todo lo que tiene que hacer es empezar por el principio y seguir hasta llegar a la Panorámica Técnica. Luego, piense en algo que

le guste para probar (como un sistema de aprendizaje, o de diagnóstico de algún tipo de problema) y extraiga los elementos necesarios para ello a partir de lo que se dice en el texto, y utilizando el índice y la Panorámica Técnica.

Otra posibilidad es que mantenga usted su ordenador conectado mientras lee el libro, y que vaya tecleando los ejemplos conforme avanza, viendo así cómo funcionan, lo que le ayudará a comprenderlo todo mejor. Por supuesto, si procede así, le costará mucho más tiempo llegar al final.

La observación más importante es que este libro no está estructurado como un libro de texto normal. Los Capítulos no constituyen entidades aisladas. De algún modo, conforme avance usted en su lectura, tendrá que volver repetidamente hacia atrás para conseguir poner en orden sus ideas. Algunas personas han comentado que, tal como está estructurado, se lee más bien como una novela que como un libro de texto. Observación interesante, que, en cualquier caso, conviene advertírsela previamente.

Antes de su publicación, leyeron este libro (por orden alfabético) Graham Beech, Phil Bradley, Bill Hudspith y Phil Manchester.

Todos ellos me hicieron algún tipo de comentario, y cada uno ha contribuido con algunas mejoras a la versión final. Lo cual ha sido muy amable por su parte.

Si alguien, una vez leído el libro, tiene cualquier comentario que pudiera conducir a modificaciones útiles en el futuro, que le envíe una nota al editor para hacérselo saber.

Chris Naylor, 1983

Nota preliminar del traductor

Que éste es un libro desmitificador se intuye por su título y queda patente tras su lectura. Su tono humorístico y desenfadado arropa, en realidad, una considerable densidad conceptual que no queda en mera descripción abstracta, sino que se plasma en programas concretos. Es, también, una excelente aproximación práctica al tema del aprendizaje, e incluye programas que ilustran cómo puede inducirse conocimiento a partir de ejemplos.

Como traductor que, además, ha analizado y probado los programas incluidos en el libro, sólo le encuentro un inconveniente: lo farragoso de tales programas (derivado, en buena parte, del uso del lenguaje BASIC), que hace verdaderamente incómodo el seguir los detalles de los algoritmos (si bien sus líneas generales quedan muy bien expuestas en el texto). Por ello, me ha parecido interesante completar esta edición en español con un apéndice en el que se dan versiones en Pascal de los dos programas más largos y más interesantes. De este modo, el lector interesado en dichos detalles y que posea unos mínimos conocimientos de Pascal, podrá analizar los algoritmos e introducir las modificaciones y mejoras que se le ocurran con mayor facilidad (aun en el caso de que quiera o de que se vea obligado a seguir codificando en BASIC).

Parte de los programas de la edición original estaban compuestos tipográficamente, y se habían deslizado algunas erratas. Para no incurrir en el mismo defecto, aquí se reproducen los listados obtenidos de la impresora. En este punto, tengo que expresar mi agradecimiento a Gonzalo Montero Ruiz y a María Teresa Galán Izquierdo, que han corregido y adaptado los programas BASIC en sus versiones para el Apple II y el Spectrum, respectivamente.

Gregorio Fernández, 1985

1. EL PORQUÉ DE LOS SISTEMAS EXPERTOS

«Se considera que un sistema experto es la incorporación en un ordenador de un componente basado en conocimiento que se obtiene a partir de la pericia de un experto, de forma tal que el sistema pueda dar CONSEJOS INTELIGENTES o tomar una DECISION INTELIGENTE acerca de una función de procesamiento. Una característica adicional que es deseable, y que para muchos es fundamental, es la capacidad del sistema para, bajo demanda, JUSTIFICAR SU PROPIA LINEA DE RAZONAMIENTO de una forma inmediatamente inteligible para el que pregunta. El estilo de programación que se adopta para conseguir estas características es la PROGRAMACION BASADA EN REGLAS».

(Definición formal de sistemas expertos aprobada por el comité del grupo especialista en sistemas expertos de la British Computer Society).

Hace mucho, mucho tiempo, cuando la tierra era aún joven y el sol salía cada mañana luciendo una gran sonrisa, no existían cosas como los Sistemas Expertos. Y, de pronto, parece que todo el mundo habla ahora sobre ellos. ¿Por qué?.

Pues bien, la respuesta es que los Científicos han dictaminado que hay una cantidad cada vez mayor de Fondos Públicos destinada a las aplicaciones de los ordenadores, y, además, también han dictaminado que hay pocas posibilidades de acceder a esta cantidad de dinero, a menos que exista una buena parte de actividad investigadora costosa y eso-

térica sobre las aplicaciones de los ordenadores. Y así, con este fin, han inventado los Sistemas Expertos, que, como nadie en la Administración sabe lo que son, permiten atraer los Fondos Públicos, aunque sólo sea para tratar de saber qué son.

Hay científicos menos mercenarios (cuyos nombres en este momento no recordamos) que no están interesados en los Fondos Públicos *per se* y que, de hecho, sufren la notable falta, hasta el momento, de Fondos Públicos para este campo. Estos altruistas simplemente dicen que quieren hacer que los ordenadores sean más accesibles a más personas. Quieren hacer que los ordenadores *piensen* como las personas. Quieren que los ordenadores *sustituyan* a las personas. Quieren que los ordenadores sean de *uso agradable*. Y, en fin, probablemente quieren también que los ordenadores asuman el cometido de la asignación de los Fondos Públicos.

Todo esto es muy bonito, pero el verdadero problema radica en averiguar sobre los Sistemas Expertos lo suficiente, como para que uno también pueda comenzar a atraer para sí mismo Fondos Públicos. Despues de esta puntualización, el libro no va a prestar mucha atención al tema de los Fondos, pero le va a permitir que, con optimismo, llegue usted a entender los Sistemas Expertos, de modo que pueda construir el suyo propio, sea subvencionado con Fondos Públicos o no.

En la primera parte del libro se construye un Sistema Experto, contando con que usted parte de una absoluta ignorancia. Se trata de un sistema de aprendizaje por medio de ejemplos, que puede recoger habilidades en un amplio abanico de áreas temáticas: *adquiere* experiencia.

Más tarde, suponiendo que se dispone de algún conocimiento en un tema concreto, construiremos un Sistema Experto capaz de utilizar de modo inteligente este conocimiento, para dar *consejos* del mismo modo que lo haría un experto humano.

Una observación que, en realidad, tenemos que hacer, es la siguiente: todos los ejemplos de este libro están en BASIC, tanto para facilitar su comprensión como para contentar a los perezosos. Pero el libro no contiene una panorámica del lenguaje ni enseña a programar con él. Este es el único elemento de experiencia que el lector, o la lectora, deberá aportar por su cuenta a la tarea.

Los ejemplos importantes se dan en BASIC, y los programas más largos están escritos en dialectos para el Apple II y el Sinclair Spectrum, de modo que pueden modificarse fácilmente para la mayoría de los micros. Los fragmentos cortos de programa están escritos en el BASIC de Apple (es decir, en Applesoft).

El capítulo final del libro resume la información dada en las páginas anteriores, de modo que, cuando llegue usted hasta allí, podrá ver qué es exactamente lo que ha aprendido con la lectura de lo anterior.

1.1 ¿PARA QUÉ QUIERE USTED UN SISTEMA EXPERTO?

La mayoría de los sistemas expertos que existen tienen dos grandes defectos, a saber: que usted, personalmente, no sabe cómo funcionan, y que usted, personalmente, no dispone de ninguno.

Estos defectos pueden, en casos extremos, llegar a ser bastante graves. Puede que usted se escabulla, eludiendo las miradas de otras personas. Que evite la conversación con ellas y se esconda tras los manuales de COBOL. Que, angustiosamente, les oiga hablar de «bases de conocimientos», «inteligencia artificial» y «representaciones del mundo real». Que no se atreva, por temor al ridículo, a acercarse inocentemente y preguntar qué es todo eso. A sus propios ojos, se convierte usted en un proscrito y una persona despreciable.

Todo ello puede ser algo fastidioso durante los primeros cinco minutos aproximadamente. Concretamente, hasta que usted intuye que el asunto no puede realmente ser tan difícil, como demuestra el hecho de que todos aquellos que pretenden comprenderlo no son, probablemente, más inteligentes que usted mismo.

Todo lo que necesita, para poner las cosas en su sitio, es que usted disponga de su Propio Sistema Experto. Un Pequeño Diseño Conocido Hecho a la Medida, le permitirá apoyarse con seguridad en la barra de cualquier bar y pontificar sobre el tema de los sistemas expertos. En lugar de esconder la cabeza y guardar su propio parecer, podrá mirar con desprecio a los que tienen sistemas expertos, pero no se los han construidos ellos mismos. Podrá reirse condescendientemente de aquéllos que no saben en realidad cómo funciona su propio sistema experto. Y podrá levantar burlonamente una ceja ante los que no tienen ningún sistema experto (imposible entender cómo se las arreglan para pasar sin él).

Para usted (el tipo alto y seguro que está en el centro del bar, arengando a un tropel de admiradores, muchos de los cuales son jóvenes y doncellas) está escrito «Construya su propio sistema experto».

No se requiere ningún conocimiento especial, aunque sería conve-

niente tener acceso a un ordenador. De otro modo, mucho de lo que sigue podría parecer, de algún modo, simplemente académico.

1.2. ¿PARA QUÉ QUIEREN OTRAS PERSONAS UN SISTEMA EXPERTO?

Los sistemas expertos tienen dos usos principales, que se corresponden con los conceptos sociológicos de función manifiesta y función latente.

La función manifiesta de los sistemas expertos consiste en proporcionar experiencia humana sobre un ordenador. Pueden, por ejemplo, diagnosticar enfermedades, deducir estructuras químicas, sugerir lugares donde extraer metales preciosos, y realizar una gran cantidad de tareas similares. En alguna medida, son de uso agradable, e incorporan el conocimiento humano de una forma más o menos parecida a la forma en que pueda retenerlo un experto humano. Y, como un experto humano, pueden incluso enseñar su experiencia a otros.

La otra función de los sistemas expertos —insinuada anteriormente— es su función latente, que consiste, como puede sospecharse, en desconcertar al ignorante con explicaciones misteriosas sobre cómo han sido construidos. Típicamente, utilizan grandes ordenadores, de los que usted no dispone, y emplean lenguajes exóticos, como LISP y PROLOG, de los que tampoco dispone usted. Esto les pone en situación de ventaja a la hora de acaparar el mercado para el suministro de sistemas expertos, porque, si la demanda puede estimularse mediante la descripción de las funciones latentes de un sistema experto, esta demanda no puede satisfacerse (¿seguro?) por alguien que simplemente tenga un micro y un intérprete de BASIC. Lo cual, sin duda, afecta al precio.

La salida a este problema es dar una guía orientada a micros para construir sistemas expertos. No una guía completa que contenga necesariamente la última palabra sobre el tema, pero sí lo suficiente para acabar con algo del misticismo y conseguir que una persona normal pueda empezar.

Hasta ahora, el verdadero problema para la comprensión de los sistemas expertos ha sido la inexistencia de un libro sencillo que acometiera una introducción al tema. A menos que esté usted preparado para hacer reflexiones bastante rigurosas sobre el asunto, todas las publicaciones actualmente disponibles parece que acentuarán la dificultad de llegar a esa comprensión, lo que induce a abandonar el tema y... ¡dejárselo a los expertos! Todo esto recuerda el intento de escalar una monta-

ña (bueno, quizá solo una colina) en la que no se encuentra un sitio donde apoyar el pie por primera vez. Con sólo poder empezar, podría usted sentirse mucho más optimista sobre lo que puede hacerse, e incluso sería capaz de llevar adelante por sí mismo muchos de los pasos siguientes sin la ayuda de un libro. Y, después de todo, esto es lo que los humanos con gran experiencia en sistemas expertos están haciendo: desarrollarlos por su cuenta.

El truco consiste en que usted mismo se ponga en la misma línea de pensamiento que siguen los pioneros actuales del tema. Que se dé cuenta de que hay algo concreto y accesible sobre lo que reflexionar, con algo donde apoyarse. Que piense que hay algunos problemas interesantes, y que esos problemas, con un poco de reflexión, se pueden resolver perfectamente, y que las soluciones que se encuentran se pueden explicar perfectamente en lenguaje llano. Que el tema de los sistemas expertos no es realmente un arte místico, sino, como todos los temas de informática, algo tan práctico y terrenal como la carpintería.

1.3 ¿QUÉ ES UN SISTEMA EXPERTO?

Antes de que se ponga la bata y se precipite hacia su mesa de trabajo, no estaría mal que se detuviera un momento y se preguntara: ¿qué es, realmente, un sistema experto?. Porque, si piensa usted en ello, se dará cuenta que la respuesta a esta pregunta podría muy bien tener un importante efecto sobre el producto final. Hoy por hoy, como veremos, la respuesta no va a afectar demasiado al producto, pero usted debería pensar en ello.

Todo comenzó hace muchos años, en la lejanísima época en que los ordenadores eran casi tan potentes como las calculadoras de bolsillo y se hablaba con respeto de los transistores. Los informáticos, aunque sobrecogidos y mudos por la potencia y la complejidad de los monstruos que habían engendrado, encontraron las palabras para expresar sus más profundos deseos.

«Sería bonito», se decían unos a otros, «si pudiésemos conseguir que este cacharro hiciera alguna otra cosa, además de los cálculos de nóminas».

«Sí», convenían, «tenemos literalmente cientos de palabras de memoria, y ya trabaja más rápido que el jefe de contabilidad, pero, sin embargo, funciona como un idiota».

«Ciertamente es un idiota», admitía otro, «pero nuestro ordenador es algo mejor que él».

Y todas las noches bebían cerveza e intentaban inventar esquemas que permitieran desatar toda la potencia de su monstruo frankensteiniano.

Bien, pues todavía permanecen allí sentados, bebiendo cerveza y conspirando contra el jefe de contabilidad, mientras el ordenador sigue haciendo la nómina. El progreso en conseguir que el ordenador haga algo más no ha sido extraordinario, pero la motivación sigue siendo la misma. Porque, como Frankenstein, lo que ellos querían era una forma de alentar algo de vida en su invento. No querían una admirada máquina sumadora, sino una verdadera máquina pensante.

Así es, desde luego, como nació todo el campo de la inteligencia artificial, y el de los sistemas expertos no es más que una parte de él.

La gente se dio cuenta de que el jefe de contabilidad no era sólo una admirable máquina sumadora (a pesar de los rumores), sino que tenía algo de experto en su propio campo. Podía, por ejemplo, falsear los libros de una forma bastante inasequible a la capacidad de cualquier ordenador.

Y no eran sólo los contables los que podían vencer a los ordenadores. En todos los campos había expertos humanos con conocimientos y habilidades especiales que los hacían indispensables y costosos. Siempre que un grupo de informáticos hablaba con alguien experto en algún campo, le escuchaban, quizá, durante un par de horas. Y luego comenzaban a suspirar, pensando en el día en que pudieran sustituirlo por un ordenador al que se pudiera desconectar y olvidarse de pagarle.

El sueño era seductor. Tan seductor que, con el tiempo, tomó la fuerza suficiente como para que la gente pensara que podría haber algún modo de hacerlo realidad. El problema era: ¿Cómo?.

Es fácil ver que, si el problema se resolviera, el resultado podría llamarse un «sistema experto». Tendría la experiencia que previamente se encontraba en los expertos humanos, y podría desconectarse durante la noche. Pero el problema de cómo conseguir tal cosa no sólo ha resultado ser difícil de resolver: apenas ha sido resuelto aún.

Pregúntele a cualquiera cómo funciona un experto humano, y, aparte de comentarios como «lentamente», observará que nadie lo sabe. O, al menos, no se sabe lo suficiente como para escribir un programa de ordenador que lo haga. Cualquier explicación contendrá palabras como «juicio» y «experiencia», que, simplemente, no se encuentran en el mundo de los lenguajes formales.

Pero las ideas básicas son bastante sencillas. Las personas, se ha

dicho, son máquinas pensantes de propósito general. Deles cualquier problema y un poco de experiencia, y podrán usar su juicio para encontrar una solución satisfactoria al problema. Una persona no es más que una colección de células cerebrales conectadas de alguna manera entre sí, y un ordenador no es más que una colección de células de memoria, también conectadas entre sí de alguna manera. Por tanto, escribamos un programa que resuelva problemas (en general), y tendremos un sistema que sustituirá a las personas. Con las ventajas adicionales de que no cae enfermo, no se olvida, no se equivoca, no pide que se le pague, y así sucesivamente. El problema es que este resolutor de problemas de propósito general ha resultado ser bastante difícil de conseguir.

Menos difíciles han sido los resolutores de problemas específicos. Por ejemplo, usted podría tener un cálculo a realizar. Bien, podría escribir fácilmente un programa que se lo hiciera, De hecho, si ha usado un ordenador, ya lo habrá efectuado. Si le da una expresión aritmética a su máquina, ésta puede calcularle el resultado. Esto parece bastante obvio, pero, al evaluar una expresión aritmética, el ordenador ha hecho todo lo que habría hecho un matemático. No se ha limitado a sumar unos cuantos números; ha tomado una cadena de símbolos y los ha manipulado hasta ponerlos en una forma razonable. Después de eso, las operaciones aritméticas han sido triviales.

Y si piensa usted que ese ejemplo no viene al caso, considere un problema que su ordenador actual no puede resolver inmediatamente. Por ejemplo, el problema de integrar una expresión matemática. La integración puede ser especialmente difícil para algunas funciones, por lo que usted podría pensar con razón que es un asunto adecuado para un sistema experto. Pero suponga que usted hubiera escrito tal sistema. ¿A qué se parecería? Se parecería, probablemente, a su ordenador actual con su intérprete de BASIC. La única diferencia estaría en que tendría algunas expresiones más en el lenguaje. De hecho, si observa usted algunas calculadoras de bolsillo, verá que tienen una tecla «integrar» que permite hacer la integración numérica de cualquier función. Por tanto, no hay nada de extraordinario en ello.

Y esto sirve para ilustrar lo siguiente: una vez que sabemos cómo hacer algo y podemos escribir un programa para hacerlo, deja por completo de parecer extraordinario. No hace mucho, si usted no era un licenciado en matemáticas, tenía que llamar a alguien para realizar una integración. En otras palabras, tenía que buscar un experto. Con las calculadoras y los programas disponibles hoy ya no es necesario, porque la tarea ha dejado de ser (virtualmente) una tarea que requiera un experto. Y, como consecuencia, difícilmente podemos llamar sistema experto a la tecla «integración» de una calculadora de bolsillo.

Por tomar un ejemplo más cotidiano, hubo en otro tiempo personas expertas en nóminas. De hecho, algunas de las mentes más ágiles del Imperio Británico podían encontrarse en el Cuerpo de Pagos del Ejército. Peritos en el descubrimiento de cuentas que no cuadran, de casos especiales, de descuentos anteriormente desconocidos, esos expertos humanos se mantuvieron en el poder durante años. Fue sencillamente el programa de nóminas el que hizo tomar a esos contables la ruta de los dinosaurios. El programa de nóminas fácilmente puede considerarse como un ejemplo de sistema experto: incorpora toda la experiencia humana en el tema de las nóminas. Pero ya nadie piensa así de él. Es, simplemente, otro caso de un problema que parece trivial una vez que se ha resuelto.

Y esto es lo que realmente pasa hoy con los sistemas expertos. Los avances realizados se han despreciado como si no lo fueran, por la sencilla razón de que, como los primeros informáticos, estamos todos sentados y reunidos deseando conseguir que el ordenador haga algo «especial». Algo más de lo que ya está haciendo. Pero, si lo consiguiéramos, aún seguiríamos reunidos rumiando nuestro descontento, porque estaríamos buscando algo extra que aún no habríamos conseguido.

Cada avance trae consigo otro programa de ordenador; ni más ni menos que eso. Pero nunca aparece el último Gran Avance que traiga el programa definitivo. Aquél que finalmente aliente vida en el monstruo.

Y, para poner peor las cosas, si tal sistema apareciese, lo que sucedería inmediatamente es que aparecería una nueva casta de expertos. Expertos humanos en el funcionamiento de los sistemas definitivos. Sabrían más sobre ellos que ninguna otra persona, y se les pagaría bien por su experiencia. Y si de entrada no lo cree usted así, pregúntese cómo aparecieron en escena los programadores.

Supongamos, por ejemplo, que se inventa un sistema capaz de realizar diagnósticos médicos. Esto, desde luego, ya se ha hecho. ¿Qué cree usted que pasaría después? Pues bien, aparecerían artículos eruditos en revistas académicas, describiendo el sistema y explicando lo que podría hacer. Y otros, observando lo que no podría hacer, emprenderían la construcción de un sistema experto mejor. Esto, por supuesto, ya ha sucedido. Es historia pasada. Podríamos resumir diciendo: se escribió un programa de ordenador, que luego fue criticado y mejorado por expertos humanos, que luego escribieron un programa mejor, y así sucesivamente...

Entonces, inevitablemente, cualquier sistema experto sólo será un paliativo temporal. Algo para retardar por algún tiempo la angustia de la privación. El truco está en encontrar un buen paliativo y no uno ma-

lo. Algo que le haga a usted pensar «¡esto es inteligente!», aunque no siempre continúe pensando que lo es. Un programa de ordenador, en suma, que haga algo que usted no se imaginara en absoluto que podría hacerse con un ordenador. Un programa de ordenador que haga algo para lo cual usted hubiera pensado que se requerían los servicios de un experto humano.

Pero siempre, a fin de cuentas, un programa de ordenador.

1.4 ¿QUÉ QUIERE HACER USTED CON SU SISTEMA EXPERTO?

Habiendo definido un sistema experto como un programa de ordenador, ni más ni menos, podríamos descansar satisfechos con el conocimiento de que ya sabíamos lo que era un sistema experto y de que ya teníamos uno. Esta es la salida fácil. La difícil es seguir el enfoque funcional y preguntar: ¿qué queremos que haga el sistema experto?.

Esta es una pregunta peligrosa, porque su respuesta puede involucrar la realización de algún trabajo, cosa que generalmente se considera desagradable. Pero para eso están los ordenadores. En general, los programas de ordenador no se definen sino en virtud de lo que hacen. Por tanto, si los sistemas expertos son programas de ordenador, ¿qué es lo que hacen?.

Y en este punto, la responsabilidad pasa, temporalmente, a usted. Porque la pregunta no es «¿qué quiere alguien, en general, hacer con un sistema experto?», sino «¿qué quiere *usted*, en particular, hacer con su propio sistema experto?». Después de todo, es usted quien va a construirlo; también podría tener algo que decir sobre el asunto, si pudiera.

Pues bien, la respuesta es, desde luego, fácil. Cómodamente sentado en su sillón, conecta usted su ordenador, cierra los ojos y sueña. Cuanto le rodea, se desvanece. Una cálida sensación de placer inunda todo su ser. Una sonrisa relajada y segura flota en sus labios, mientras perezosamente teclea la pregunta: ¿COMO PUEDO HACERME MILLONARIO?. Y, a modo de respuesta, en la pantalla aparecen unas cuantas frases sencillas y lógicas.

«¡Naturalmente!», exclama usted. «Sí, claro que sí. Es evidente que, de ese modo, me haré millonario. Lástima no haber construido antes mi sistema experto».

Y apunta usted la respuesta, y luego pasa a interrogar a la máquina

sobre la anteriormente fastidiosa cuestión de cómo evitar que su cabello siga encanecido, y luego sobre cómo conseguir que no vuelvan a crecer malas hierbas en su césped.

Todo es como un sueño, y, en efecto, es un sueño.

Lo cual es una pena, verdaderamente. Pero este tipo de historias y la falta de programación conducen a ello.

El problema es que algunas cosas son imposibles de programar, sean sistemas expertos o no, y si quiere usted programar algo imposible, entonces va a encontrar dificultades para llevarlo a la realidad.

El retorno de los sueños de grandeza al mundo real, en el que está usted sentado con un ordenador enfrente, no tiene por qué ser demasiado decepcionante.

Decir que algunas cosas son imposibles de programar no quiere decir que todo sea imposible de programar. Pero puede que usted se pregunte a qué viene tanto hablar de programación, si lo que realmente quiere es avanzar en la construcción de su propio sistema experto. Bien, sólo queremos asegurarnos de que la expresión «sistemas expertos» no se confunde con «panacea universal». Tomemos un caso análogo que es conocido desde hace un poco más de tiempo: las bases de datos.

Las bases de datos son cosas bastante complejas. Pueden requerir una cantidad grande de conocimientos y de desarrollo. Ello puede conducir a una especie de parálisis intelectual en aquellos que se encuentran por primera vez con las bases de datos. Y, sin embargo, si usted tiene unos cuantos ficheros (¿y quién no los tiene?), entonces tiene una base de datos (en cierto modo). Por tanto, ¿dónde está el problema?. Para ser francos, no hay ningún problema. Todo consiste en introducirse de modo gradual en el asunto, y en poco tiempo se consigue llegar a comprenderlo.

Lo mismo ocurre con los sistemas expertos. Realmente, no tienen grandes misterios. Lo cual es una buena cosa, teniendo en cuenta que está usted pensando en construir uno.

Hay una idea que, sin embargo, conviene tener en mente: la idea de un sistema experto como un sistema con «juicio». Ahora bien, todos los programas de ordenador tienen alguna cantidad de juicio. Siempre que escribe usted unas líneas de programa para comparar un valor con otro y tomar una determinada acción dependiendo del resultado, está introduciendo juicio.

Lo que ocurre es que el juicio juega un papel central en el diseño de los sistemas expertos. El juicio, más que el cálculo, tiende a ser característica típica. Pero como el juicio normalmente se obtiene como resultado de un cálculo, la diferencia tiende a ser conceptual más que verdaderamente real.

Por tanto, piense en algo que le gustaría que hiciera un sistema experto, y pregúntese si puede reducirse a una sucesión de juicios. Si es así, tiene usted una buena ocasión para construir un sistema experto que lo haga. Y aquí hay una gama de posibilidades extraordinariamente amplia.

Para ayudarle a empezar, he aquí algunas posibilidades:

Diagnóstico de enfermedades comunes.
Localización de fallos en circuitos sencillos, o, incluso, en un televisor.
Diagnóstico de enfermedades de plantas.
Registros electrocardiográficos.
Clasificación de animales, aves o plantas, según su especie.

Pero seguramente tendrá sus propias ideas.

1.5 SISTEMAS EXPERTOS: ALGUNAS FALSEDADES

Sobre el tema de los sistemas expertos suelen decirse muchas cosas, y no todas ellas son objetivamente ciertas.

El error más común, aunque parezca raro, no consiste en exagerar lo que un sistema experto podría, en principio, hacer, sino en subestimar sus posibilidades.

Típicamente, usted podría escuchar que «un sistema experto sólo puede hacer...», seguido de una lista de lo que puede hacer, sobreentendiéndose que no puede hacer ninguna otra cosa. Esto es absurdo. Si usted puede pensar en una forma de hacer algo, entonces puede construir un sistema experto que lo haga.

El sistema experto de alguna otra persona puede que sólo sea capaz de hacer tal y tal cosa. ¡Pero el suyo! Este es otro asunto. El suyo no tiene tales restricciones. Va a ser más grande y mejor que cualquier otro (si es así como lo construye, naturalmente). Cualquiera que le diga qué es lo que no puede hacer su sistema experto, es un presuntuoso.

Consideremos, por ejemplo:

«UN SISTEMA EXPERTO SOLO PUEDE SER EXPERTO EN UNA COSA». Evidentemente falso. Todo lo que tiene usted que hacer es construir uno que sea experto en dos cosas y habrá demostrado que el crítico estaba equivocado.

«UN SISTEMA EXPERTO SOLO PUEDE HACER LO QUE PODRIA HACER UN EXPERTO HUMANO (EN EL MEJOR DE LOS CASOS)». También falso. Supongamos que usted decide construir un sistema para algo sobre lo que no existieran expertos humanos. Si de algún modo llegara a funcionar, habría demostrado que esta afirmación era falsa. Más típica es la existencia de muchos campos en los que la experiencia humana dista bastante de ser completa. En tales casos, todo lo que su sistema experto tiene que hacer es funcionar un poco mejor que el juicio humano normal.

«LOS SISTEMAS EXPERTOS JAMAS REEMPLAZARAN AL HOMBRE». Naturalmente que lo harán. Si no, no tendría ningún sentido construirlos.

Siendo como es la naturaleza humana, siempre será posible encontrar a alguien que no esté de acuerdo con la descripción de sistemas expertos que se da en este libro. Incluso usted mismo puede discrepar. Por tanto, permítasenos hacer las siguientes observaciones:

Para algunas personas, una característica especial de los sistemas expertos es que son «adaptativos», lo que significa que el comportamiento del sistema cambia (normalmente para mejor) a lo largo del tiempo. Y eso pueden hacerlo de dos maneras: o bien guardando su conocimiento en reglas que el usuario puede alterar fácilmente, o bien construyendo su conocimiento a partir de su propio análisis de las entradas, con poca participación del usuario. En ambos sentidos, los sistemas que se describen en este libro son adaptativos.

Un propósito en la construcción de muchos de los sistemas expertos actuales es hacerlos expertos generales en una amplia gama de tareas, estableciendo, sencillamente, una distinción entre el conocimiento que utilizan y los mecanismos que manipulan ese conocimiento. Al cambiar la naturaleza del conocimiento específico, el sistema experto puede ejercer su capacidad de manipulación para hacerse experto en un nuevo campo. Por ejemplo, un sistema experto diseñado para realizar diagnóstico médico, podría, quitando el conocimiento médico y sustituyéndolo por conocimiento relativo a ingeniería estructural, hacerse experto en ingeniería estructural. Los programas que se describen en este libro están organizados así.

Se dice, con frecuencia, que un característica especial de los sistemas expertos es la organización basada en reglas de sus instrucciones. A diferencia de los programas de ordenador convencionales, que siguen un sistema de ejecución línea a línea, los sistemas expertos consisten en una colección de reglas que no se ejecutan secuencialmente, sino que «se activan» sólo cuando se dan las condiciones apropiadas. En sentido literal, esta descripción es algo engañosa, porque, aparte de máquinas muy raras, todos los ordenadores avanzan paso a paso a lo largo de su programa en lenguaje de máquina. Sin embargo, conceptualmente es razonable hacer esa observación, pero sólo en el sentido de que los sistemas expertos se comportan a menudo como si cada línea de programa comenzara con la palabra IF... Y así es, quizás indirectamente, como funcionan los programas que se dan en este libro.

2. UN ESQUEMA ESTADÍSTICO

2.1 ELABORACIÓN DE UNA MATRIZ

Pero, basta ya de preliminares, puede que piense usted. ¿Dónde está el sistema experto prometido? ¿Cómo, en suma, va usted a construir su propio sistema experto?

Pues bien, es fácil.

Para empezar, defina un «array» bidimensional, y considérelo como una matriz rectangular. Piense en las preguntas que quiere usted plantearle a su sistema experto, y etiquete las columnas de la matriz con todas las posibles respuestas. Piense luego en los elementos de información que el experto podría necesitar para poder llegar a esas respuestas, y etiquete las filas de la matriz con esos elementos de información.

Tomemos un ejemplo concreto, para que quede más claro.

Supongamos que le gustaría que su sistema experto le dijese si va a llover mañana. La idea es que usted quiere llegar a poder inclinarse sobre su ordenador y preguntarle: «¿va a llover mañana?». Y el ordenador pensará durante un momento y responderá «Si» o «No». Para evitar confusiones, no deberá responder «Si y No».

Hay, pues, dos posibles respuestas. Por tanto, necesitamos una matriz con dos columnas. Todavía no sabemos cuántos elementos de información va a necesitar el experto para responder a la pregunta, pero supongamos, por ejemplo, diez filas en la matriz. Por tanto, en BASIC, haremos DIM E(10,2). Por si no lo ha adivinado, la E es por experto. Gráficamente, tendremos:

Fig. 2.1

Una matriz de experto

Observación	Mañana lloverá	Mañana no lloverá
1	a	b
2	c	d
.	.	.
.	.	.
.	.	.
10	s	t

Donde, a la izquierda y de arriba hacia abajo, hemos listado los elementos de información sobre el tiempo. Son los números de 1 a 10, y podrían corresponder a preguntas como: ¿llueve hoy?, ¿hace frío hoy?, y así sucesivamente. Las letras a, b, ...s, t en la matriz, son elementos que todavía no tenemos, pero que representarán la posibilidad de que cada resultado sea cierto. Más adelante los rellenaremos con información relativa al tiempo. La tarea del sistema experto es decidir cuál de las dos columnas, Mañana Lloverá o Mañana No Lloverá, hay que seleccionar, dadas las respuestas a las preguntas 1 a 10.

Por ejemplo, la observación 1 podría darse cuando la respuesta fuera SI a la pregunta «¿Hace frío hoy?». De numerosas observaciones (realizadas por los expertos) sobre los días fríos, se sabe que hay un 60 por 100 de posibilidades de que mañana llueva. Por tanto, «a» será 60, ó 0,6, si prefiere los decimales. De manera análoga podríamos obtener valores para b, c y así sucesivamente, que garantizaran la consecución de una respuesta razonable, basada en las observaciones que hacemos.

En este momento, el asunto podría ser algo más concreto de lo que es, pero, aparte de eso, ¿hemos hecho realmente algo útil?. Pues bien, sí. Hemos establecido una Base de Conocimientos y un Dominio de Consultas, expresiones ambas interesantes de recordar para la próxima vez que le apetezca enrrollarse en un bar.

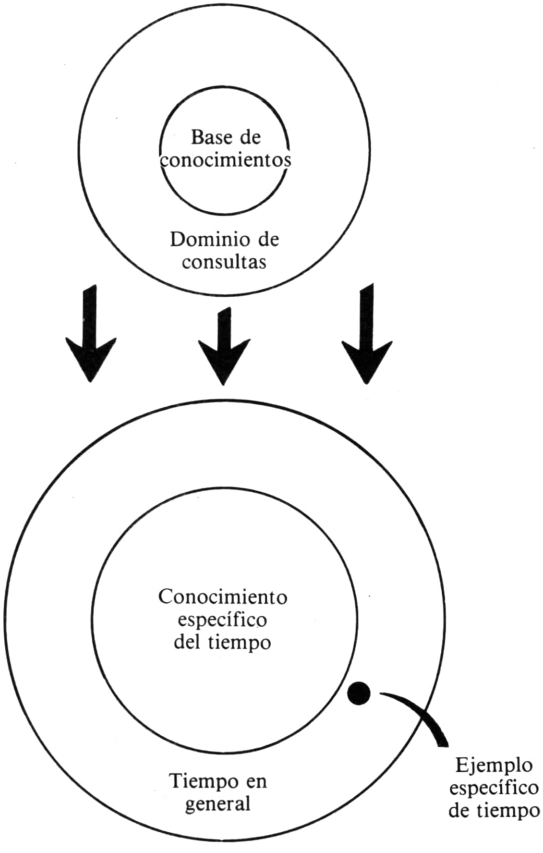
Consideremos el Dominio de Consultas. Pues bien, eso es todo lo que hay. Literalmente. Es el asunto del que entiende el sistema experto. Si alguna cosa cae dentro del Dominio de Consultas, entonces puede usted preguntar sobre ella a su sistema experto, y si no, no puede. Así de fácil. En este caso, el Dominio de Consultas es el tiempo (o la lluvia, si quiere usted restringirlo un poco). No tendría ningún sentido preguntarle a este experto cómo hacerse millonario, porque tal pregunta quedaría fuera de su Dominio de Consultas. Puede haber otras razones por las que no puede hacerse esa pregunta, pero, por el momento, basta con una.

Consideremos ahora la Base de Conocimientos. En este caso, es la matriz E(10,2), que contiene todo el conocimiento del experto sobre el tema, y que está vacía, lo cual es significativo. No sabe una palabra sobre el tiempo. Lo que significa que, de momento, tampoco tiene ningún sentido preguntarle a este experto sobre el tiempo.

Esto es crucial para nuestro tema principal, de modo que echémosle un vistazo a la Figura 2.2.

El Dominio de Consultas es el «campo» en el que se intenta que el Sistema Experto sea experto. En este ejemplo tenemos un experto en pronósticos meteorológicos, por lo que el Dominio de Consultas es El Tiempo.

Fig. 2.2
Base de conocimientos y dominio de consultas



Para ser experto en este campo, el sistema experto necesita una Base de Conocimientos, que es la información que le damos sobre el tema del tiempo. En un mundo ideal (que, en nuestro caso, no existe) la Base de Conocimientos cubriría todo el Dominio de Consultas, es decir, el sistema sabría todo cuanto puede saberse sobre el tema en cuestión (El Tiempo). En la práctica, casi nunca se sabe todo; sólo se conoce una parte. Podemos representar este hecho mostrando la Base de Conocimientos más pequeña que el Dominio de Consultas, pero incluída dentro de él.

Cuando surge una pregunta específica, esa pregunta debe caer dentro del Dominio de Consultas, si es que queremos tener alguna esperanza de respuesta. Sin embargo, puede que caiga o que no caiga dentro de la Base de Conocimientos en nuestro diagrama. El lugar que corresponde exactamente a un ejemplo específico, depende de que la Base de Conocimientos dentro del Sistema Experto cubra o no ese ejemplo específico.

Consideremos un experto humano al que se dirige usted con una aguda pregunta relativa a la probabilidad de que llueva mañana. La mayoría de las personas se inclinarían a aventurar algún tipo de suposición sobre el asunto. Pero ellos tienen, hasta hoy, una ventaja sobre el ordenador: saben lo que es el tiempo, y pueden mirar por la ventana para ver cómo es hoy, lo que podría darles una pista sobre el tiempo que hará mañana. Obviamente, queremos que nuestro sistema experto haga algo similar.

Distinguiremos, por tanto, dos clases diferentes de información en el problema. Aunque pueden denominarse de muchas formas, las llamaremos Información Fija e Información Variable.

La Información Fija es la que estaría comprendida dentro de la Base de Conocimientos. Contiene datos invariables que, en este caso, serían sobre el tiempo en general.

La Información Variable es la que no está comprendida dentro de la Base de Conocimientos. Es información específica referente al problema concreto. En este caso sería la información específica que le permitiría al experto decir si lloverá mañana o no, supuesto que estamos planteando la pregunta hoy, y que no estamos preguntando sobre cualquier día en general.

En cierto sentido, la diferencia entre esos dos conjuntos de información es la misma que hay entre programa y datos. La información fija es parte del programa, y la información variable son los datos para este problema específico. Pero, aunque sólo sea para asegurarnos de que el asunto no queda demasiado claro, vale la pena hacerlo de nuevo confuso, señalando que podríamos querer alterar la información fija bastante a menudo (con lo cual no va a ser realmente fija) y que a veces podría,

incluso, ser tan variable como la misma información variable. Esto es debido a que la mayoría de los expertos cambian sus métodos de trabajo conforme pasa el tiempo, y no quisiéramos impedir que nuestro sistema experto pudiera aprender a lo largo del tiempo, ¿verdad?. De hecho, ésta es frecuentemente una característica de los sistemas expertos.

Volvamos, en cualquier caso, al experto en el tiempo meteorológico. Le hemos preguntado si va a llover mañana, y recibimos como respuesta que no tiene ni idea. ¿Y luego, qué?.

Pues bien, démosle algo para que pueda funcionar. Enseñémosle algo sobre el tiempo. Por ejemplo: si hoy está lloviendo es más probable que llueva mañana. Esto es así, porque la lluvia tiende a venir por rachas. Hay rachas de lluvia y rachas sin ella. Por tanto, si hoy está lloviendo, es probable que haya una racha de lluvia, lo que implica que mañana lloverá. Lo cual, a propósito, es más o menos cierto. Así, lo que estamos haciendo es incorporar un poco de experiencia en nuestro sistema experto.

Supongamos que estimamos que si hoy está lloviendo hay un 60 por ciento de posibilidades de que mañana llueva. Por consiguiente, por medio de un maravilloso juego de manos aritmético, habrá un 40 por ciento de posibilidades de que no llueva mañana. Supongamos también que, si hoy hace un tiempo seco, hay un 55 por ciento de posibilidades de que mañana tampoco llueva. Por medio de la misma hazaña aritmética, calculamos que habrá un 45 por ciento de posibilidades de que llueva. Volvamos ahora a escribir nuestra matriz original con estas joyas de conocimiento metidas en ella:

	1. Mañana lloverá	2. Mañana no lloverá
1. Lluvioso	60	40
2. Seco	45	55

Llegados a este punto, resulta bastante evidente lo que tenemos que hacer. El sistema experto tiene que imprimir la afirmación «Mañana Lloverá» o la afirmación «Mañana No Lloverá».

Para decidir cuál es, el sistema preguntará si llueve hoy. Si es así imprime «Mañana Lloverá», porque es el resultado más probable. Si no, puede hacer una de dos cosas, dependiendo de cómo quiera usted programarlo:

Puede imprimer «Mañana No Lloverá», suponiendo que si no llueve hoy es que el tiempo es seco, y, por tanto, es más probable que mañana también sea seco. O bien puede preguntar específicamente si hoy hace un tiempo seco y, al recibir la respuesta «Si», proceder con ella.

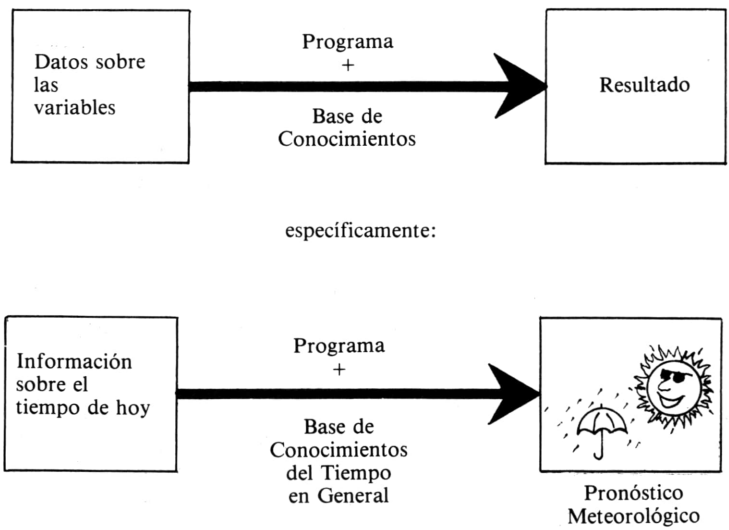
Puede que ahora todo esto sea concreto, pero ¿es útil?. Después de todo, usted siempre puede atender al hombre del tiempo, o, si no puede, no tiene más que esperar y ver qué tiempo hace. Tampoco es tan importante, podría usted pensar.

Y, lo que es más importante, probablemente no tenga usted mucha confianza en esos números que hemos puesto para empezar en la matriz (perdón, Base de Conocimientos).

Bien, es razonable tener algunas reservas. Pero consideremos algunos aspectos de lo que hemos hecho.

Tomemos esos números que nos hemos inventado. Ciertamente, son algo arbitrarios. Pero supongamos que no lo hubieran sido. Supongamos que hubieran sido números determinados con precisión. ¿Qué pasaría entonces?. Pues bien, en determinadas circunstancias, podría haber algunas diferencias. Supongamos que uno de los números hubiera sido 100 por ciento, y el otro 0 por ciento. Entonces, nuestro sistema experto podría haber hecho una decisión segura, no sólo una conjetura. Más aún, supongamos que no conociéramos en absoluto esos números. Podríamos no saber que la lluvia típicamente precede a la lluvia, o que el tiempo va por rachas. Si el sistema experto, en vez de nosotros, pudiera haber llegado a esas conclusiones, entonces sabría más de lo que nosotros inicialmente sabíamos, y podría predecir el tiempo mejor que nosotros mismos.

Fig. 2.3
Pronóstico del tiempo



Consideremos ahora la pregunta real planteada: ¿va a llover mañana?. Probablemente bastante trivial para el super-culto sistema experto con el que intentamos impresionar al mundo. Pero podría haber sido una pregunta diferente. Las filas y columnas de la matriz podrían haberse etiquetado de otro modo, con diferentes variables, diferentes resultados y un campo de experiencia diferente.

Podríamos haber preguntado: ¿va el Barça a ganar la liga esta temporada?, o ¿indica mi dolor de cabeza que tengo resaca?.

Supuesto que nos las podemos ingeniar para especificar algunas variables, algunos resultados y algún método para enlazar unas con otros, podemos atacar una gran variedad de problemas con nuestro sistema experto. El truco está, simplemente, en disponer de algún método en torno al problema.

2.2 PROBABILIDADES

En el ejemplo anterior, razonábamos que un suceso tenía, por ejemplo, el 55 por ciento de posibilidades de ocurrir, el 45 por ciento de posibilidades de ocurrir, el 100 por ciento de posibilidades de ocurrir. No hay ninguna razón especial por la que no podamos seguir así, pero el uso de probabilidades presenta algunas ventajas.

Las probabilidades son algo muy preciso, y se representan por números comprendidos en el rango de 0 a 1.

Si es absolutamente cierto que un suceso ocurre, entonces tiene una probabilidad de ocurrencia igual a 1. Si un suceso no tiene absolutamente ninguna posibilidad de que ocurra, entonces tiene una probabilidad de ocurrencia igual a cero. Si las posibilidades de que ocurra o no ocurra un suceso son mitad y mitad, entonces su probabilidad de ocurrencia es igual a 0.5. Y todos los otros casos estarán entre 0 y 1.

Consideremos los casos de 0 y 1. Hay una diferencia entre una probabilidad de exactamente 1, y una probabilidad que es aproximadamente 1, o sea, tan cercana a 1 que podemos apuntarla como 1. Si un suceso tiene una probabilidad de 1 exactamente, entonces ese suceso *tiene que* ocurrir. Sencillamente, no puede no ocurrir. Puede tener un número de probabilidad asociado a él, pero no hay en absoluto ningún elemento de azar en él. Tiene casi la fuerza de una sentencia causal. Considere lo siguiente: si suelta usted su vaso cuando no hay nada bajo él, entonces caerá al suelo. Este es un suceso seguro, que tiene asociada una probabilidad de 1. Y es también una sentencia causal, en cuanto que la causa de la caída del vaso es que usted lo ha soltado.

Pero considere esto otro: cuando, después de un tiempo, se va usted del bar en el que estaba con un vaso, descubre que todos los demás se van también. En resumen, todos salen al mismo tiempo con probabilidad 1: es un suceso seguro. Pero no es un suceso estrictamente causal. No es el hecho de que usted, personalmente, salga, lo que causa que todos los demás decidan que el bar ya no tiene atractivo para ellos. El elemento causal es el hecho de que el camarero está cerrando y despidiendo a todos.

Por tanto, si dos sucesos ocurren conjuntamente con probabilidad 1, la relación entre ambos no tiene por qué ser estrictamente causal; podría ser simplemente que siempre se dieran juntos. Pero siempre van juntos, y no hay duda sobre ello.

Volvamos ahora al bar y dejemos caer de nuevo el vaso. Decíamos que caería al suelo, pero no decíamos que realmente fuera a romperse. Probablemente será así, y quizás se romperá con una probabilidad 0.999999. Para ahorrarnos escribir todos esos nueves, podríamos decir que se romperá con una probabilidad de aproximadamente 1. Pero esto no es lo mismo que decir que se romperá con una probabilidad de exactamente 1. Si la probabilidad hubiera sido exactamente 1, entonces el vaso tendría que romperse sin que cupiera otra posibilidad. Pero, en la medida en que no se llega a esta mágica cifra 1, siempre hay alguna probabilidad, aunque sea pequeña, de que el suceso no ocurra. Y la diferencia sería un vaso no roto.

Los mismos argumentos valen para la probabilidad 0. Un suceso con probabilidad 0 *no puede* ocurrir. Pero un suceso que tiene probabilidad de aproximadamente 0, puede ocurrir, aunque es muy raro.

Puede que ya se haya dado cuenta de que no hay ninguna razón especial por la que no podamos utilizar porcentajes. Una posibilidad de ocurrencia del 100 por ciento significa que es seguro que algo ocurre. Con una certidumbre menor, podríamos tener 99.99 por ciento, y así sucesivamente.

En realidad, usted puede elegir y utilizar cualquier escala. Podría tener una escala de menos cinco a más cinco, si quisiera. U otra cualquiera. La única razón para insistir en esto es dejar claro que es posible hablar sobre probabilidades de una manera muy precisa. Para nuestros propósitos, un suceso completamente causal puede describirse mediante números de probabilidad, si queremos hacerlo así. Y la razón para utilizar el rango usual de probabilidades de 0 a 1 sólo es para recordar que lo que estamos haciendo es preciso.

Tomemos nuestra sentencia «si llueve hoy, entonces hay un 60 por

ciento de posibilidades de que llueva mañana». Queremos decir, naturalmente, que hay una probabilidad de ocurrencia 0.6 para el suceso. Ahora bien, de ahí podemos deducir que, si observásemos un número infinito de días lluviosos, encontraríamos que 0.6 de entre ellos irían seguidos por otro día lluvioso. Sin condiciones ni dudas. En este caso, desde luego, nuestra cifra es casi con seguridad imprecisa, pero eso no quita sentido a la sentencia en la que se ha usado esa cifra. El significado era muy preciso, sólo la cifra estaba equivocada. Y si hubiésemos tenido la cifra correcta, la sentencia no sólo habría tenido significado; también habría sido cierta.

Y ésta es la razón para insistir sobre el asunto.

Imagínese a sí mismo hablando sobre su sistema experto. «Mi sistema ha decidido —proclama usted— que hay una probabilidad 0.75 de que estalle la Tercera Guerra Mundial en uno cualquiera de los días de la próxima semana».

«Ya», responde una audiencia defraudada. «¿No está seguro?». Y se ponen a hablar sobre algo menos incierto. Como el tiempo, por ejemplo.

Sería absolutamente irritante tener un sistema experto capaz de calcular algo así y verlo rechazado tan alegremente. Son las cosas que verdaderamente sacan a uno de quicio.

Pero fíjese usted en lo que ha dicho. Estaba usted usando una medida de probabilidad, de modo que, si hay una probabilidad 0.75 de que estalle la Guerra Mundial Versión 3, en uno cualquiera de los días de la próxima semana, entonces la probabilidad de que no estalle en cualquier día de la próxima semana es $1 - 0.75 = 0.25$. Y para que no haya guerra durante toda la semana, es preciso que no estalle en ninguno de sus siete días, a lo que corresponde una probabilidad 0.25^7. Con lo que resulta que la probabilidad de una Semana de Paz es 0.000061035.

Por consiguiente, la probabilidad de que no tengamos una Semana de Paz será $1 - 0.25^7 = 0.999938965$. Que corresponderá a la probabilidad de que la Tercera Guerra Mundial estalle en cualquier día de la próxima semana.

En todo lo anterior se supone, desde luego, que cuando estalla la Guerra Mundial 3, lo hace de manera totalmente independiente de lo que pudiera haber ocurrido en los días precedentes, pero como hoy todo está controlado por ordenadores, esta suposición parece razonable: después de todo, los chips tienden a fallar de una manera estadísticamente independiente.

Usted no ha especificado en qué día de la próxima semana va a ocurrir, pero hay menos de 7 posibilidades entre 100.000 de que no ocurra en ninguno. Por tanto, resumiendo, cualquier persona sensata, basándose en la fuerza de la información que acaba usted de darle, saldría a comprar cascos de acero.

Pero lo importante a recordar es que no hay por qué despreciar las probabilidades. Se pueden calcular muchas cosas exactas con ellas y hacer afirmaciones muy exactas, algunas de las cuales podrían llegar a ser ciertas.

Y, desde un punto de vista más práctico, si usted va a construir su propio sistema experto, no se sentiría animado a llegar muy lejos, si pensara que cualquier cosa con una probabilidad es sólo una vaga «posibilidad» que no merece en absoluto la pena de ser considerada en serio.

2.3. MÁS PROBABILIDADES

Ahora bien, podría ocurrir que las probabilidades fueran, por así decirlo, un libro abierto para usted. Por el contrario, por si no lo fueran, puede que valga la pena mencionar unos pocos puntos básicos sobre lo que sucede cuando tenemos más de una probabilidad.

Alguna notación podría ayudar. Intentémoslo escribiendo P(A), P(A&B), P(A:B), que definiremos del siguiente modo:

P(A) es la probabilidad de que ocurra el suceso A.
P(A&B) es la probabilidad de que los sucesos A y B ocurran ambos.
P(A:B) es la probabilidad de que ocurra el suceso A, dado que el suceso B ha ocurrido.

Respectivamente, diremos que:

P(A) es la probabilidad de A.
P(A&B) es la probabilidad CONJUNTA de A y B.
P(A:B) es la probabilidad CONDICIONAL de A, dado B.

Y no son tres maneras diferentes de decir la misma cosa.

Volviendo a nuestro ejemplo del tiempo, teníamos la matriz (utilizando probabilidades en lugar de porcentajes):

	1. Mañana Lloverá	2. Mañana No Lloverá
1. Lluvioso	0.6	0.4
2. Seco	0.45	0.55

Se trata de probabilidades CONDICIONALES. Son las probabilidades de que mañana haga un tiempo particular, DADO el tiempo que hace hoy. Por ejemplo, hay una probabilidad 0.6 de que llueva mañana si hoy llueve. Esto no es lo mismo que la probabilidad de que llueva mañana, ni la probabilidad CONJUNTA de que llueva mañana y llueva hoy. A primera vista podría parecer que fueran la misma cosa. Pero no lo son, y vale la pena aclarar el asunto enseguida, porque, de otro modo, aunque todas esas historias que hemos contado sobre probabilidades son precisas, podrían utilizarse mal.

En primer lugar, ¿cuál es la probabilidad de que llueva mañana?.

Observe que no hemos dicho nada acerca del tiempo que hace hoy. Es una pregunta sin relación con el tiempo del día precedente. En general, ¿va a llover mañana?. La información de que disponemos es que, si un día cualquiera está lluvioso, entonces la probabilidad de que llueva es 0.6. Y, por otra parte, si está seco, la probabilidad de que llueva el día siguiente es 0.45. ¿Cómo hemos de hacer para obtener la probabilidad total de que llueva mañana?. ¿Sumándolas (1.05)?. ¿Promediándolas (0.525)?.

Sea P(M) la probabilidad de que llueva mañana, y utilicemos L y S para denotar Lluvioso y Seco hoy, respectivamente. Queremos encontra P(M), y nuestra tabla nos da P(M:L) y P(M:S) en la primera columna.

En este momento, introducimos una fórmula:

$$P(A\&B) = P(A:B) * P(B).$$

Dicho con palabras, la probabilidad de que A y B ocurran ambos es igual a la probabilidad de A dado B, multiplicada por la probabilidad de B.

Si piensa usted en ello, encontrará que es totalmente razonable. Queremos saber la probabilidad de que A y B ocurran. Si sabemos que B ocurre, entonces hay una probabilidad P(A:B) de que A ocurra también. Pero B sólo ocurre con probabilidad P(B), lo que hemos de tener en cuenta para obtener la cifra correspondiente a que ambos, A y B, ocurran. Por tanto, hacemos el producto de P(A:B) por P(B), que, como P(B) es inferior a 1, tiende a hacer más pequeña la respuesta al considerar la posibilidad de que B no ocurra.

Por tanto, la probabilidad de que mañana llueva y que hoy llueva es:

$$P(M\&L) = P(M:L)*P(L)$$

es decir, la probabilidad de que llueva hoy y mañana es igual a la probabilidad de que mañana llueva, dado que hoy llueve, multiplicada por la probabilidad de que llueva hoy.

O bien: $P(M\&L) = 0.6*P(L)$
Y si el día de hoy es seco:
$P(M\&S) = P(M:S)*P(S)$
O bien: $P(M\&S) = 0.45*P(S)$.

Pero buscábamos P(M), la probabilidad de que mañana llueva. Pues bien, la probabilidad de que mañana llueva es, evidentemente, igual a la probabilidad de que mañana y hoy llueva, más la probabilidad de que mañana llueva y hoy no llueva. Por tanto:

$$\begin{aligned} P(M) &= P(M\&L) + P(M\&S) = P(M:L)*P(L) + P(M:S)*P(S) = \\ &= 0.6*P(L) + 0.45*P(S) \end{aligned}$$

En este momento debemos recordar que o bien hoy es un día lluvioso, o bien es seco, por lo que $P(L) + P(S) = 1$, es decir, es una cosa u otra con probabilidad 1. Por tanto:

$$\begin{aligned} P(M) &= 0.6*P(L) + 0.45*(1-P(L)) = \\ &= 0.6*P(L) + 0.45-0.45*P(L) = \\ &= 0.15*P(L) + 0.45 \end{aligned}$$

Por consiguiente, para poder responder a la pregunta «¿cuál es la probabilidad de que llueva mañana?» necesitamos saber la probabilidad de que hoy sea un día lluvioso.

Esta es, desde luego, una pregunta de fácil respuesta, porque suponemos que la probabilidad de que llueva mañana es la misma probabilidad de que llueva cualquier otro día, lo que significa que es la misma probabilidad de que llueva hoy. Es decir $P(M) = P(L)$, y, por tanto:

$P(M) = 0.15*P(M) + 0.45$;
$0.85*P(M) = 0.45$;
$P(M) = 0.45/0.85 = 0.53$, aproximadamente.

Lo que nos dice que hay que esperar más bien a que mañana llueva, lo mismo que ha ocurrido hoy y seguirá ocurriendo hasta el fin del mundo. Así es el tiempo en Inglaterra.

Observe que no podríamos haber llegado a este resultado si P(M) no hubiera sido lo mismo que P(L). Si P(L) hubiera correspondido, por ejemplo, a que hoy hubiera niebla, no sería igual a la probabilidad de que mañana llueva. Hubiésemos tenido, entonces, que recurrir a alguna otra fuente para encontrar un valor de P(L).

Lo importante a destacar es que la probabilidad de que mañana llueva no está presente en aquella tabla original que teníamos. Aquellas eran probabilidades CONDICIONALES, no probabilidades de sucesos aislados, que son cosas totalmente distintas.

En el transcurso de los cálculos anteriores también habíamos obtenido las probabilidades CONJUNTAS P(M&L) y P(M&S), que valían 0.6*P(L) y 0.45*P(S), respectivamente. Ahora bien, sabemos que P(M) = P(L) y que P(S) = 1-P(L) = 1-P(M), con lo que resulta:

P(M&L) = 0.6*P(M) = 0.3176, aproximadamente, y
P(M&S) = 0.54*(1-P(M)) = 0.2117, aproximadamente.

Estas probabilidades son mucho más pequeñas, porque corresponden a que llueva hoy y llueva mañana (es decir, dos días lluviosos consecutivos) y a que llueva hoy y mañana no (es decir, dos días con distinto tiempo). Estas probabilidades conjuntas representan demandas más exigentes para situaciones específicas. Al suponer menos cosas, hay menos posibilidades de que ocurra aquello por lo que estamos preguntando.

Con toda probabilidad, usted estará ya harto de probabilidades. Verdaderamente, no constituyen una lectura particularmente interesante, debido a la casi total ausencia de argumento, cadáveres y chistes.

Sin embargo, hay una cosa más a decir en este Apartado. Tras ello volveremos al sistema experto propiamente dicho.

El 16 de julio de 1917, en 2.5 horas, cayeron nada menos que 118 litros por metro cuadrado de lluvia en Kensington. El 18 de agosto de 1924, en cinco horas, 203 litros cerca de Bridgewater. El 18 de junio de 1917, 243 litros durante un día en Bruton, Somerset.

Y, trasladándonos a otras partes del Imperio: En Gibraltar, el 25 de octubre de 1836, 765 litros en un solo día. En Bagnio, Filipinas, durante los días comprendidos entre el 14 y el 17 de julio de 1911, la lluvia caída fue: 889 litros, 736 litros, 432 litros y 203 litros.

Todo lo cual sirve para demostrar que hay algo que aún no hemos incluído en nuestro sistema experto: el conocimiento de que a veces ya no llueve: diluvia.

2.4. MÁS VARIABLES

Tenemos nuevamente un día típico. Hace frío, llueve, sopla el viento y hay niebla. Se sube usted el cuello de su gastada gabardina para protegerse de los elementos, y enciende un cigarrillo para entrar en calor. Una gota de agua de tamaño más que regular le apaga el cigarrillo.

«¿Seguirá así mañana?», se pregunta a sí mismo.

Pues bien, por supuesto que probablemente seguirá así. Cualquiera podría decirle que estamos en Inglaterra, la tierra donde cualquier cosa es posible, siempre y cuando implique que la lluvia le empape a uno.

Pero, aunque sólo sea para dar rienda suelta a una descabellada fantasía, hay que admitir que podría darse un día seco. Por tanto, ¿va a llover mañana?.

Conecte el sistema experto para averiguarlo. Antes de conectarlo, compruebe que no haya entrado agua de lluvia dentro de la carcasa. ¿Recuerda la matriz E(10,2)?. Pues bien, rellénela del siguiente modo:

	1.Mañana lloverá	2.Mañana no lloverá
1. Frío	P(Lluvia:Frío)	P(Seco:Frío)
2. Lluvioso	P(Lluvia:Lluvioso)	P(Seco:Lluvioso)
3. Viento	P(Lluvia:Viento)	P(Seco:Viento)
4. Niebla	P(Lluvia:Niebla)	P(Seco:Niebla)

Supongamos que conoce usted las probabilidades condicionales y puede rellenar fácilmente la matriz. Si, por ejemplo, hoy hiciera frío y NO Lloviera ni hiciera Viento ni Niebla, entonces debería estar contento (es una manera de hablar, claro). Porque todo lo que tendría que hacer sería mirar en la tabla y ver si la probabilidad condicional de Lluvia dado Frío es mayor o menor que la probabilidad condicional de No Lluvia (Seco) dado Frío, y escoger el resultado más verosímil: aquel que tiene mayor probabilidad.

Sin embargo, surge la pregunta de qué hacer cuando hay más de una variable. En este caso, tenemos cuatro. ¿Cómo va a calcular el sistema experto si mañana lloverá o no?.

Puede parecer una pregunta trivial. Después de todo, si verdaderamente hace frío, viento, niebla, y llueve, parece que cualquier idiota podría predecir lluvia con un razonable grado de certidumbre. Parece como una racha de mal tiempo que ha de continuar. Y eso es lo que verdaderamente importa.

Si hasta un idiota puede hacer una conjetura aceptable sobre la base de esa información, su sistema experto debería ser capaz de hacer una conjetura al menos tan buena como ésa.

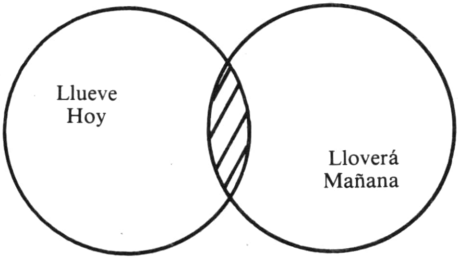
Pensemos primero en las variables que tenemos:

Hace frío. Esto suena a mal tiempo, pero ¿significa lluvia?. Realmente, a menudo no significa lluvia. En tiempo frío, la atmósfera no puede contener tanto vapor de agua como en tiempo cálido, de modo que, si lloviera, el agua que habría en las nubes preparada para caer no sería mucha. Para apoyar esta observación, puede consultar las cifras mensuales de precipitación en Gran Bretaña. Descubrirá que no llueve mucho más en los meses de invierno que en los de verano. Lo cual puede que le sorprenda, si guarda usted vivos recuerdos de haber quedado calado hasta los huesos en invierno. Probablemente sea que no recuerde la lluvia cálida tan bien como la fría.

Llueve. Bien, el tiempo tiende a venir por rachas, de modo que un día lluvioso podría fácilmente preceder a otro día lluvioso. A menos, desde luego, que no haya quedado agua en las nubes para caer mañana. (Esta última suposición se conoce con el nombre de Optimismo).

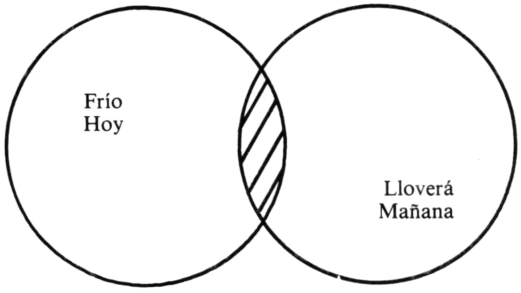
Hace viento. Esto suele preceder a la lluvia. El aire cálido y húmedo, al ser soplado hacia una zona fría, va a elevarse y a enfriarse, y va a llover a cántaros sobre nuestras cabezas. Y los movimientos de aire como ése son otra forma de describir un día de viento.

Fig. 2.4
Dos variables correlacionadas



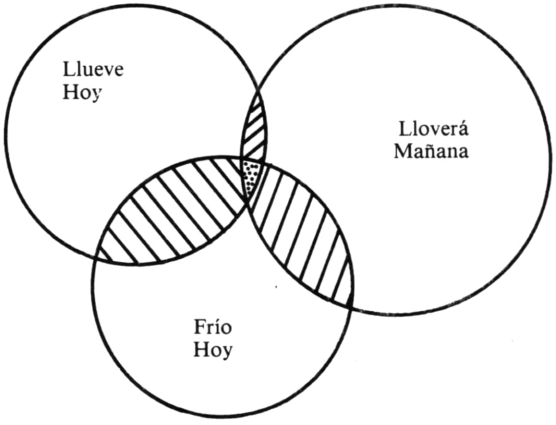
La figura 2.4 representa dos variables, Llueve Hoy y Lloverá Mañana, y la zona rayada indica que esas dos variables están correlacionadas. Si sabemos si Llueve Hoy o no, podemos estimar la probabilidad de que Llueva Mañana, observando en qué medida esas dos variables «se solapan».

Fig. 2.5



Similarmente, podríamos proceder del mismo modo, si una de nuestras variables fuera Frío Hoy en lugar de Llueve Hoy.

Fig. 2.6
El caso de tres variables

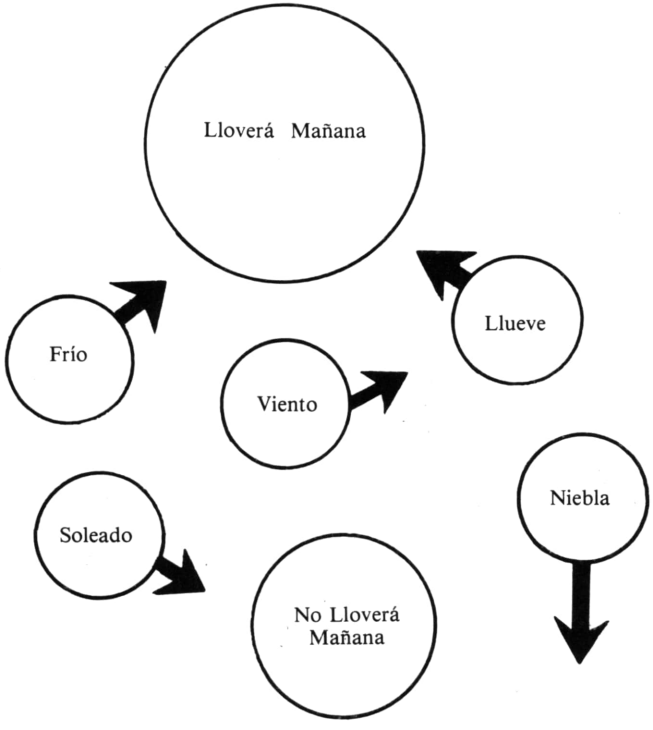


Sin embargo, si tenemos la información de Llueve Hoy y Frío Hoy simultáneamente, el problema se hace mucho más difícil, porque el tamaño del solapamiento sobre Lloverá Mañana depende del solapamiento que existe entre Llueve Hoy y Frío Hoy, es decir, del grado de correlación entre esas dos variables. Esto es lo que muestra la figura 2.6.

Hay niebla. Bien, si hay niebla no es probable que haga viento, ¿verdad?. La niebla normalmente es debida al aire húmedo inmóvil. Puede que se ponga a llover o que no.

Por tanto, tomando nuestras cuatro variables, ¿Qué es lo que tenemos?. Hasta este momento, nada, muy cierto. Algunas de ellas, pero no todas, apuntan lluvia. Ciertamente, sería fácil si todas ellas indicaran lluvia con una probabilidad mayor que 0.5, porque, en ese caso, no dudaríamos en hacer que nuestro sistema experto predijese lluvia. Pero supongamos que, en lugar de Niebla, hubiéramos utilizado la variable Soleado. Y supongamos que el tiempo hubiera sido: frío, lluvioso, con viento y soleado. Ahora bien, esto, que es lo que los meteorólogos llaman «tiempo inestable», para nosotros es un fastidio, porque Soleado tiende a indicar que mañana hará un tiempo seco, mientras que todas las otras variables tienden a indicar lluvia. En general, tenemos un conjunto de punteros, y todo va bien si todos ellos apuntan en la misma dirección.

Fig. 2.7
Punteros al Tiempo



Si tenemos varios indicadores, podemos conseguir que nuestro sistema experto encuentre fácilmente el resultado, si todos los indicadores apuntan en la misma dirección. Si apuntan en distintas direcciones, nos encontramos con problemas, porque no sabemos cuánta confianza hay que depositar en cada uno.

Por tanto, ¿qué hay que hacer?.

Volvamos a nuestras probabilidades condicionales; comenzábamos con P(Lluvia:Lluvioso): la probabilidad de que Llueva Mañana, supuesto que hoy está lloviendo.

Pero lo que hemos de obtener es P(Lluvia:suceso X), donde X es el suceso correspondiente a que hoy esté frío, lluvioso, con viento y con niebla, es decir, queremos calcular:

P(Lluvia: Frío & Lluvioso &Viento & Niebla)

y, concretamente, queremos saber si es mayor o menor que:

P(No Lluvia: Frío & Lluvioso & Viento & Niebla).

Volviendo a una de las fórmulas que teníamos antes:

P(Lluvia:suceso X) = P(Lluvia & suceso X) / P(suceso X)

o bien:

$$P(\text{Lluvia : Frío \& Lluvioso \& Viento \& Niebla}) =$$

$$= \frac{P(\text{Lluvia mañana\&Frío hoy\&Lluvioso hoy\&Viento hoy\&Niebla hoy})}{P(\text{Frío \& Lluvioso \& Viento \& Niebla})}$$

Ahora se comprenderá un poco mejor por qué, anteriormente, hemos dedicado cierto tiempo a hablar de los diferentes tipos de probabilidades: para dejar claro que, acerca de una situación, podemos tener diferentes tipos de información y plantear diferentes tipos de preguntas.

Por ejemplo, habíamos comenzado con declaraciones como P(Lluvia:lluvioso). Lo cual no está exactamente en la misma forma que la pregunta que acabamos de considerar. No se puede, en general, partir de un montón de declaraciones simples sobre probabilidades condicionales, y combinarlas para formar directamente una gran declaración sobre la probabilidad de que ocurra un suceso, dada la ocurrencia conjunta de un montón de otros sucesos.

En el caso de que no esté usted convencido, considere un ejemplo.

Hay bruma y niebla. La probabilidad de que llueva mañana, supuesto que hoy hay bruma, es 0.75, y esa misma es la probabilidad de que llueva, supuesto que hay niebla. Como hay bruma y niebla, sumamos ambas posibilidades y obtenemos una probabilidad 1.5 de que llueva

mañana, lo cual es evidentemente falso: las probabilidades no pueden ser mayores que 1.

Pero supongamos que consideramos P(A&B), probabilidad de dos sucesos conjuntamente, siendo el primer suceso que llueva mañana, supuesto que hoy hay bruma, y el segundo que llueva mañana, supuesto que hoy hay niebla. Supongamos que pudiéramos decir que ambos sucesos son mutuamente independientes, con lo que tendríamos P(A&B) = P(A)*P(B), que, en este caso, sería igual a 0.75 por 0.75, es decir, 0.5625.

Pues bien, eso es falso: teniendo dos sucesos que apuntan lluvia, hemos calculado una probabilidad de lluvia que es inferior a la que da cada uno de los sucesos.

La respuesta es, naturalmente que la bruma y la niebla son lo mismo. El origen de ambas son gotitas de agua en el aire, y la única diferencia es que las gotitas son más grandes en la niebla que en la bruma.

Así, niebla y bruma no son más que dos palabras diferentes que designan la misma cosa, y, al incluir las dos declaraciones, no hemos dicho, en absoluto, nada adicional sobre la situación.

Por consiguiente, la probabilidad de que llueva mañana es 0.75: la misma predicha por cada una de las declaraciones.

Y es fácil ver que en nuestra tabla original no había absolutamente nada que diera cuenta de esto. Si tuviésemos Niebla y Bruma como entradas en el lado izquierdo de la tabla con las correspondientes probabilidades condicionales asociadas, tendríamos realmente la misma entrada duplicada, pero nada hay que le indique al sistema experto que podría ocurrir tal cosa.

Hasta cierto punto, siempre vamos a tropezar con este mismo problema, con mucha diversidad de formas diferentes según sean las variables, porque lo que hemos encontrado es una correlación entre las variables. Si todas ellas fueran independientes entre sí, la vida sería más fácil, pero, en general, no lo van a ser. Consideremos Lluvia & Viento. Estas dos variables, ciertamente, no representan en absoluto la misma cosa, pero, cuando llueve, también suele hacer viento: las variables no son independientes. Si hubiésemos incluído Soleado, tampoco esta variable habría sido independiente de Niebla, porque, cuando una está presente, la otra tiende a no estarlo.

Afortunadamente, hay una forma de abordar este problema. Lamentablemente, requiere algún esfuerzo. Lo que tiene usted que hacer es lo siguiente:

1. Considere todas las variables que tiene; sean n en total.
2. Piense en todas las combinaciones posibles entre esas variables, cuyo número es la suma de «x elementos tomados de entre n», para x = 0 hasta n, es decir:

$$\frac{n!}{(n-0)!*0!} + \frac{n!}{(n-1)!*1!} + \frac{n!}{(n-2)!*2!} + \quad \ldots\ldots \quad + \text{ y así sucesivamente}$$

En el caso de las cuatro variables sobre el tiempo, tendremos:

$$\frac{4!}{(4-0)!} + \frac{4!}{(4-1)!*1!} + \frac{4!}{(4-2)!*2!} + \frac{4!}{(4-3)!*3!} + \frac{4!}{(4-4)!*4!}$$

lo que da:

$$1 + 4 + 6 + 4 + 1 = 16$$

Es decir, son 16 combinaciones diferentes que podrían darse con nuestras variables (Si/No).

El signo de exclamación significa «factorial», y también es como un grito de angustia computacional.

3. Haga una nueva lista de variables, en la que deberá añadir a las propias variables todas las combinaciones posibles entre ellas.

4. Asigne a cada resultado una probabilidad condicionada a la ocurrencia de cada una de las combinaciones posibles de variables.

5. Cuando quiera utilizar el sistema, no tiene más que buscar en la tabla ampliada la combinación de variables que sea, y leer la probabilidad de cada uno de los posibles resultados.

6. Una vez que haya meditado sobre todo lo anterior, puede usted levantar las cejas, fruncir los labios, arrugar la frente y exclamar: «¡todo esto debe ser una broma!». (Signo de exclamación, no de factorial).

Pues bien, sí, lo siento, pero eso es lo que tiene usted que hacer. Algo laborioso, hay que admitir...

Naturalmente, sería mucho más fácil si no tuviese usted tantas variables. En el fondo, es culpa suya, por pensar que puede echarle cualquier cosa al ordenador, simplemente por el hecho de disponer de una gran cantidad de memoria.

2.5 EL TEOREMA DE BAYES

Sin embargo, si quiere tratar de aplicar este método, aún puede aturdirse algo más, introduciendo el teorema de Bayes en sus cálculos. El Reverendo Bayes, como puede adivinarse por su título, dedicó su vida al estudio de cosas eternas, como, por ejemplo, los problemas de estadística. Y su teorema establece que:

$$P(M:X) = \frac{P(X:M)*P(M)}{P(X:M)*P(M) + P(X:\text{no } M)*P(\text{no } M)}$$

En otras palabras, considere que M sea el suceso Mañana lloverá, y que X sea una combinación particular de sucesos que describen el tiempo que hace hoy (por ejemplo, lluvioso, con viento, frío y horrible).

Entonces, la probabilidad de «Mañana lloverá», dado X hoy, es igual a la probabilidad de X hoy, supuesto que va a llover mañana, multiplicada por la probabilidad total de que X ocurra, llueva mañana o no.

Si no ve usted muy claro por qué eso ha de ser cierto, considere lo siguiente:

$$P(M:X) = \frac{P(M \,\&\, X)}{P(X)}$$

Este es un resultado que ya habíamos obtenido antes. Entonces, tendremos tambien:

$$P(X:M) = \frac{P(X \,\&\, M)}{P(M)}$$

Por tanto, $P(X\&M) = P(X:M)*P(M) = P(M:X)*P(X)$, de donde obtenemos:

$$P(M:X) = \frac{P(X:M)*P(M)}{P(X)} = \frac{P(X:M)*P(M)}{P(X:M)*P(M) + P(X:\text{no } M)*P(\text{no } M)}$$

Ahora bien, si usted conociera P(M:X), podría introducirla desde el principio sin ningún inconveniente. Pero podría resultar más fácil conocer P(X:M); cuál de ellas va a ser más fácil, depende, hasta cierto punto, de la naturaleza de la pregunta. Para una pregunta hay que dar la probabilidad de que llueva mañana, dado X hoy. Para la otra, hay que dar la probabilidad de que ocurra X hoy, supuesto que se sabe que va a llover mañana (o, para ser más realistas, la probabilidad de que haya ocurrido X ayer, supuesto que ha llovido hoy). También hay que dar

(para que no crea usted que ya está resuelto el problema) P(X:no M), la probabilidad de que haya ocurrido X ayer, supuesto que hoy es un día seco. Y muy bien puede ser que opine usted que este método es incluso peor que cualquiera de los anteriores.

Lo cual no es del todo verdad. Ciertamente, es bastante malo si todos los sucesos están correlacionados entre sí, en cuyo caso hay que dar los valores de las probabilidades para todas y cada una de las combinaciones posibles. Pero, si son independientes, entonces se puede calcular la probabilidad global P(X:M) con mucha mayor facilidad que la probabilidad global P(M:X).

Supongamos, por ejemplo, que X consiste en la pareja de sucesos Y y Z. Entonces: P(X:M) = P(Y&Z:M) = P(Y:M)*P(Z:M), una simple multiplicación.

Pero P(M:Y&Z) no es lo mismo que P(M:Y)*P(M:Z). Si no lo ve, haga los dos cálculos por separado, y verá qué resultados tan diferentes dan.

Más adelante (mucho más adelante, en la segunda mitad del libro) se da un programa que utiliza este método para construir un sistema experto, y que funciona suponiendo que todos los sucesos son totalmente independientes entre sí (si no lo fueran, el sistema funcionaría mal). Con frecuencia, se diseñan sistemas que presuponen independencia entre las variables, sólo para hacer las cosas más fáciles. Ciertamente, se alivia el problema del ingente número de combinaciones que, en otro caso, habría que considerar, pero, a menudo, las hipótesis de independencia sólo son aproximadamente ciertas. Y, si hay mucha correlación entre las variables, el problema se hace verdaderamente inmenso.

Si alguna vez llega usted al final de este libro, quizás haría bien en retroceder hasta este Apartado y pensar de nuevo en todo esto, una vez que haya visto cómo puede realizarse un sistema utilizando este método de trabajo. O, quizás, cuando llegue al final, podría vender el libro y gastarse lo que consiguiera en unas copas. De ese modo, será menos probable que le importe si va a llover mañana o no.

N. del T. No quisiéramos aturdir aún más al lector medio, pero sí tranquilizar a aquél que tenga algunos conocimientos previos sobre la teoría de probabilidades. Porque quizás éste haya observado una aparente incorrección en el desarrollo de la última parte de este Capítulo. En efecto, parece como si el autor identificara la independencia entre variables aleatorias con la ausencia de correlación entre ellas, lo que, en general, es falso: la independencia es un requisito más fuerte; si dos variables aleatorias son independientes, entonces la correlación entre ambas es nula, pero la inversa, en general, no es cierta. Ahora bien, hay dos casos particulares en los que la correlación nula entre variables aleatorias implica su independencia. Uno es aquél en que ambas variables siguen una distribución de probabilidad normal (gausiana). Otro, cuando las variables son binarias. Y este último es justamente el caso de las variables que se consideran en este Capítulo: Lluvioso (sí o no), Frío (sí o no), etc.

3. CÓMO ELUDIR LAS PROBABILIDADES

Antes de llegar hasta aquí, una extraña desazón tiene que haberle invadido. Usted creía que construir un sistema experto sería algo fácil. De hecho, más o menos se le había prometido que así sería. Usted tiene un ordenador; ¿Por qué no puede éste hacer algo del trabajo?. ¿A qué viene, pregunta usted, toda esta historia de las probabilidades?. ¿Por qué, por ejemplo, hay que comunicarle al experto la probabilidad del más pequeño resultado, para conseguir que tome una decisión?.

Con franqueza, si tuviera usted que hacer todo ese trabajo, entonces prácticamente no necesitaría un ordenador. Podría ahorrarse tiempo y hacerlo usted mismo.

Y, en cualquier caso, si usted conociera las probabilidades asociadas con cada posible conjunto de sucesos, no necesitaría un experto mecánico. El motivo por el que usted quiere tener un sistema experto es para que le haga cosas que usted no puede hacer por sí mismo. Que le diga cosas que usted previamente no conocía.

Alguien puede pensar que todo esto no son más que buenas palabras. Sin embargo, su propio sistema experto va a ser diferente de la mayoría de los sistemas expertos actuales en el siguiente sentido:

La mayoría de los sistemas expertos, hasta hoy, se basa en una larga investigación que, toscamente hablando, implica la exploración del cerebro de un experto para encontrar las posibilidades de todos y cada uno de los diferentes resultados para todos y cada uno de los posibles sucesos. El experto (humano) conoce esas posibilidades. Las personas que exploran su cerebro y escriben un sistema experto también llegan a conocerlas, y, tras la necesaria programación, el sistema termina, asimis-

mo, por conocerlas. El sistema se entrega entonces a alguien que *no conoce* todas las posibilidades y que, como era de esperar, queda impresionado.

Algunos sistemas expertos trabajan de un modo algo diferente, en el sentido de que es el ordenador quien explora el cerebro del experto, pero en casi todos los casos se supone la presencia de un experto (humano) en el tema.

Pero considere su caso, en el que es usted quien construye su propio sistema experto. Si usted supiera todo sobre el problema, no necesitaría un sistema experto que le dijera nada sobre tal problema. Normalmente, usted no sabe todo lo que se puede saber. Y, como no lo sabe, no puede programarlo. Y, si se tomara usted el trabajo de averiguar todo acerca del tema en el que quiere experiencia, representaría una gran cantidad de trabajo adicional el escribirlo en un programa, en lugar de, simplemente, tomar nota mental de ello.

Resumiendo, sus requisitos para un sistema experto son muy diferentes de los de las personas que hasta hoy han construido sistemas expertos.

Usted quiere algo que sea de construcción razonablemente rápida y fácil, que dé buenos resultados y que no le obligue a pasarse media vida estudiando el tema sobre el que busca experiencia.

Y así es como vamos a proceder. Como esbozábamos más arriba. Justo al empezar, decíamos que nuestro sistema experto sería mejor que cualquier otro, o, al menos, diferente. Y aquí comienzan a aparecer las diferencias.

3.1 CÓMO HACER QUE EL ORDENADOR SE ENCARGUE DEL TRABAJO PESADO

Ahora, el sistema cambia. Ya no le tiene usted que dar a la máquina probabilidades (aunque ya sabe lo suficiente sobre probabilidades para valorar lo que se puede hacer con ellas). Sólo le dará la información suficiente para que la máquina pueda obtener las probabilidades por sí misma. No sólo se trata de obtener un sistema experto, sino también un sistema que aprenda, puesto que es la máquina, y no usted, quien ha de aprender. Esto es lo que se llama Delegación.

Y también es hora ya de que tenga usted un programa para introducir en el ordenador, y así convencerse de que realmente está haciendo algo.

He aquí lo que vamos a hacer: Construiremos el sistema experto de modo tal que pueda seguir una sesión de entrenamiento durante la cual sea capaz de aprender a tomar una decisión en base a la experiencia acumulada. Tras haber sido entrenado, lo dejaremos en libertad y le permitiremos tomar decisiones reales, utilizando su juicio experto. A menos que analice detenidamente el programa, usted no tiene por qué saber cómo toma sus decisiones. Por supuesto, no tiene que enseñarle explícitamente a hacerlo, como antes.

Tengo la esperanza de que ello le haga sentirse un poco más optimista sobre todo este asunto.

3.2 EL SISTEMA DE APRENDIZAJE

El programa que vamos a utilizar es el listado en la Figura 3.1, y su ordinograma aparece en la Figura 3.2. Está escrito en BASIC para el Apple II y el Sinclair Spectrum.

Si introduce este programa y lo ejecuta, resulta que hace lo siguiente:

Pregunta cuántas variables tiene usted, y DIMensiona dos vectores. Uno es para guardar una regla de juicio que él desarrollará por sí mismo. El otro es para guardar la información presente sobre las variables en un caso específico. Pregunta los nombres de las variables, para poderle hablar sobre ellas de modo inteligible. Pregunta los nombres de los dos posibles resultados, Q1$ y Q2$. Entra entonces en un bucle en el que pregunta los detalles de ejemplos específicos («¿ocurre esta variable?» —sí o no). Una vez recopilados los detalles de un caso, hace una conjetura sobre lo que debería ser, y sugiere un resultado. Si está usted conforme con el juicio, contesta S (de SI), y el sistema pasa a considerar otro caso. Si discrepa usted, el sistema modifica ligeramente su regla de juicio y pasa a otro caso.

Lo que deberá usted observar es que, gradualmente, la máquina va mejorando su comportamiento en cuanto a averiguar el resultado correcto, aunque el grado de mejora depende bastante de los ejemplos que le dé.

Fig. 3.1
Un programa de aprendizaje sencillo

Listado para el Apple II:

```
10  HOME : INPUT "CUANTAS VARIABLES
    HAY? ";V
20  DIM V(V),R(V),V$(V)
30  FOR I = 1 TO V:V(I) = 0:R(I) = 0
    : NEXT I
40  PRINT "INTRODUZCA LOS NOMBRES DE
     LAS VARIABLES"
50  FOR I = 1 TO V: INPUT "NOMBRE DE
     VARIABLE? ";V$(I): NEXT I
60  PRINT "INTRODUZCA LOS NOMBRES DE
     LOS RESULTADOS"
70  INPUT "RESULTADO 1: ";Q1$
80  INPUT "RESULTADO 2: ";Q2$
90  FOR I = 1 TO V:V(I) = 0
100  PRINT "VARIABLE ";V$(I)
110  INPUT "ES CIERTA ESTA VARIABLE?
     ";R$
120  IF R$ = "S" THEN V(I) = 1
130  NEXT I
140 D = 0
150  FOR I = 1 TO V
160 D = D + V(I) * R(I)
170  NEXT I
180  IF D >  = 0 THEN  PRINT "EL RES
    ULTADO ES ";Q1$: INPUT "ES CORRE
    CTO? ";R$: IF R$ = "S" THEN  GOTO
    90
190  IF D < 0 THEN  PRINT "EL RESULT
    ADO ES ";Q2$: INPUT "ES CORRECTO
    ? ";R$: IF R$ = "S" THEN  GOTO 9
    0
200  IF D >  = 0 AND R$ = "N" THEN  FOR
    I = 1 TO V:R(I) = R(I) - V(I): NEXT I
210  IF D < 0 AND R$ = "N" THEN  FOR
    I = 1 TO V:R(I) = R(I) + V(I): NEXT
    I
220  GOTO 90
```

Listado para el Sinclair Spectrum

```
  10 CLS :  INPUT "Cuantas variables tiene
s? ";v
  20 DIM q$(20): DIM r$(20):  DIM  v(v):
DIM r(v): DIM v$(v,20)
  30 FOR i=1  TO v:  LET v(i)=0:  LET r(i)
=0:  NEXT i
  40 PRINT AT 16,0;"Por favor,  da nombres
 a estas  variables"
  50 FOR i=1  TO v:  INPUT "Nombre de la v
ariable? ";v$(i):  NEXT i
  60 PRINT AT 16,0;"Por favor, da nombres
a los     resultados            "
  70 INPUT "resultado 1? ";q$
  80 INPUT "resultado 2? ";r$
  90 CLS :  FOR i=1  TO v: LET v(i)=0
 100 PRINT "Variable: ";v$(i)
 110 INPUT "Es cierta esta variable? ";a$
 120 IF a$="s"  THEN  LET v(i)=1
 130 NEXT i
 140 LET d=0
 150 FOR i=1  TO v
 160 LET d=d+v(i)*r(i)
 170 NEXT i
 180 IF d>=0  THEN :  PRINT "El resultado
es ";q$: INPUT "Es correcto?";a$: IF a$="s
"  THEN  GO TO 220
 190 IF d<0  THEN :  PRINT "El resultado e
s ";r$:  INPUT "Es correcto?";a$: IF a$="s
" THEN  GO TO 220
 200 IF d>=0  AND a$="n"  THEN :  FOR i=1
 TO v: LET r(i)=r(i)-v(i): NEXT i
 210 IF d<0  AND a$="n"  THEN :  FOR i=1
TO v: LET r(i)=r(i)+v(i): NEXT i
 220 CLS : PRINT AT 16,0;"Cualquier tecla
para continuar"'"BREAK para parar"
 230 LET a$=INKEY$: IF a$="" THEN  GO TO 2
30
 240 GO TO 90
```

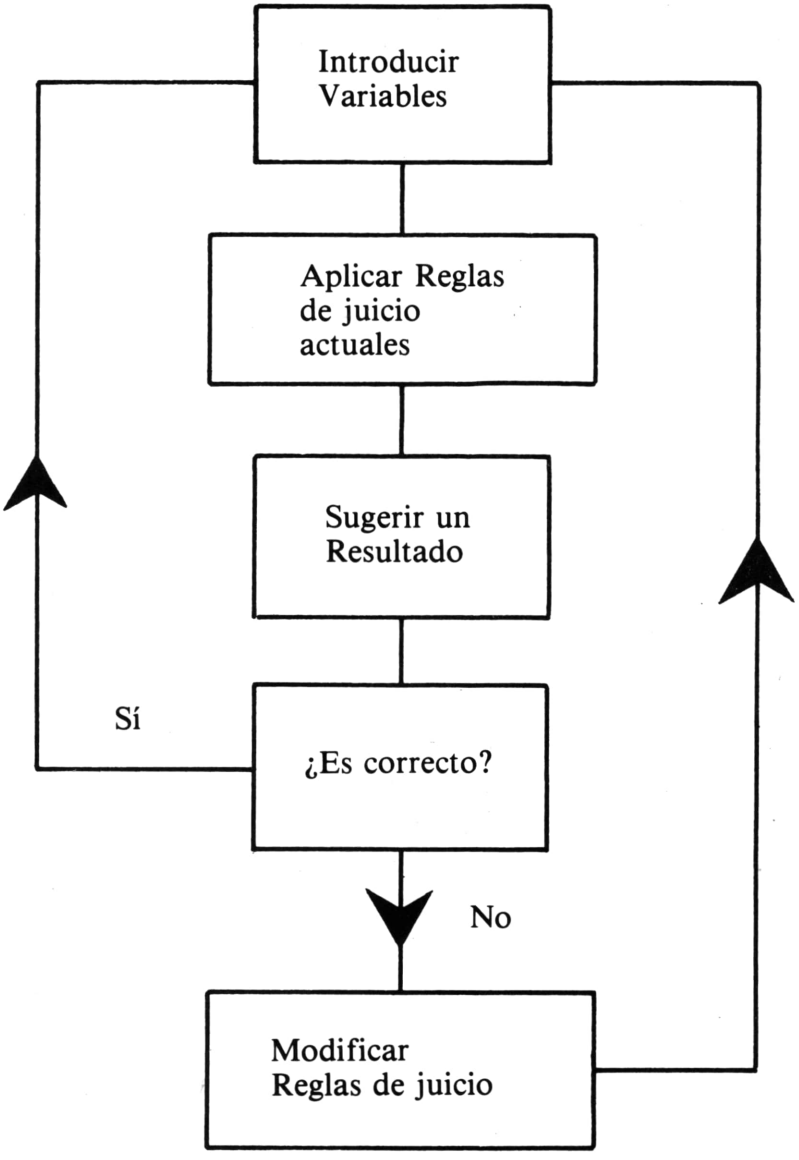


Fig. 3.2
Un proceso de aprendizaje para desarrollar un conjunto de reglas de juicio

Si, por ejemplo, usted todavía está interesado en el tiempo, y los resultados son Lloverá/No Lloverá, es difícil ver cómo podría decirle a la máquina si ha acertado o no, a menos que esté dispuesto a esperar hasta mañana para saberlo. Y un dato al día no permite una gran rapidez de aprendizaje.

Un esquema mejor consistiría en partir de una colección de registros meteorológicos, de modo que se sepa realmente cuál debería ser el resultado.

Por otra parte, no necesariamente hay que abordar ese tipo de problema. Usted podría idear algo más exacto, como clasificar objetos, por ejemplo. Suponga que piensa en un objeto que tiene que pertenecer a una de dos clases. Considere unas variables que pudieran asociarse con él, e introdúzcalas. El programa debe ser capaz de aprender a distinguir la clase a la que pertenece el objeto en el que está usted pensando.

Por ejemplo, puede que quiera usted que el «experto» le diga si se trata de un pájaro o de un avión (un típico y laborioso problema). Su lista de variables podría entonces contener elementos como: alas, cola, pico, motor, plumas, tren de aterrizaje, y así sucesivamente. Ahora bien, evidentemente, en el momento en que usted especifique, por ejemplo, que tiene un tren de aterrizaje, es bastante verosímil que sea un avión y no un pájaro, y como cada vez hay que especificar cada una de las variables, no hay forma de eludir la pregunta sobre el tren de aterrizaje. Por tanto, la respuesta es muy segura, y se trata de ver si el sistema experto puede llegar a ser tan seguro como usted mismo.

Si tomamos el problema del pájaro y el avión, y ejecutamos el programa, encontraremos que la primera vez, si estamos pensando en un avión, el sistema dice que es un pájaro. Le comunicamos que no es correcto, y a la siguiente vez acierta. La primera vez que pensamos en un pájaro, el sistema se equivoca; se lo decimos, y a la siguiente ocasión acierta. Después de lo cual, ya no vuelve a equivocarse más. Se ha convertido en un experto en distinguir entre pájaro y un avión.

Eso es admirable. Nos da la sensación de que por fin hemos conseguido algo. Pero antes de salir a celebrarlo, vale la pena dedicar algún tiempo a comprobar lo que ha sucedido, y por qué ha sucedido. Porque lo que realmente queremos saber es si este programa es o no es una solución universal para todos nuestros problemas referentes a los sistemas expertos.

En primer lugar, reseñemos las variables, variables que se ponen a «1» siempre que son ciertas:

	¿Sí?
Alas	1
Cola	1
Pico	1
Motor	0
Plumas	1
Tren de aterrizaje	0

Dicho de otro modo, el vector de variables, V, está formado por unos y ceros, según que el caso actual (u objeto bajo examen, como quiera usted llamarle) tenga o no la propiedad particular que corresponde a cada variable.

En el caso de un pájaro, V es el vector (1,1,1,0,1,0).

En el caso de un avión, V es (1,1,0,1,0,1).

Miremos ahora en el programa, y consideremos el vector R, que es la regla que el experto ha desarrollado para distinguir entre las dos posibilidades.

Encontramos que, cuando el sistema ha dejado de equivocarse, el vector R es (0,0,1,—1,1,—1).

El valor de la variable D se obtiene multiplicando los vectores R y V, de modo que:

D = D + R(I)*V(I) para todos los valores de I (de 1 a 6).

Así, si tomamos el V correspondiente a un pájaro, tendremos:

D = 0 + 0 + 1 + 0 + 1 + 0 = 2.

Y si tomamos el V correspondiente a un avión:

D = 0 + 0 + 0-1 + 0-1 = -2.

De modo que el experto puede decir que se trata de un pájaro si D es positivo (mayor que cero), y de un avión si D es negativo (menor que cero).

Como esto es todo lo que puede decirse con estas variables sobre pájaros y aviones, es evidente que el sistema experto ha aprendido correctamente y ya no pude cometer ningún error más.

Observando de nuevo el vector de regla, vemos que tiene dos ceros

en las posiciones correspondientes a las dos primeras variables, alas y cola, por lo que tales variables no se tienen en cuenta al tomar una decisión: D comienza teniendo el valor 0, y esas variables no influyen en nada para que cambie. Lo cual es bastante razonable, porque si tiene alas y cola puede ser lo mismo un pájaro que un avión; ese conocimiento no nos dice nada.

Por otra parte, las posiciones correspondientes a pico y plumas tienen ambas el valor 1, mientras que las de motor y tren de aterrizaje tienen el valor —1.

Por tanto, parece que el comportamiento del sistema es razonable, al menos en este ejemplo.

Pero, en este momento, podría parecer que estamos haciendo algo muy distinto de lo que hacíamos cuando hablábamos de probabilidades. ¿Será así?. Es bien fácil comprobarlo. Podemos escribir de nuevo la tabla, rellenándola con las probabilidades condicionales, como antes:

Pájaro		Avión	
P(Pájaro:Alas)	0.5	P(Avión:Alas)	0.5
P(Pájaro:Cola)	0.5	P(Avión:Alas)	0.5
P(Pájaro:Pico	1	P(Avión:Pico)	0
P(Pájaro:Motor)	0	P(Avión:Motor)	1
P(Pájaro:Plumas)	1	P(Avión:Plumas)	0
P(Pájaro:Tren)	0	P(Avión:Tren)	1

Si tiene alas o cola, la tabla no nos dice nada acerca de si es un pájaro o un avión (las posibilidades son las mismas para ambos), pero habrá de ser uno u otro. Las otras cuatro variables nos dicen que se trata de un pájaro o un avión, pero no de ambos, y las probabilidades asociadas son 1 y 0, lo que significa sucesos seguros. En términos gráficos, vea la Figura 3.3.

En este ejemplo, si tiene pico, entonces debe tener plumas, y no puede tener ni motor ni tren de aterrizaje, y es un pájaro con probabilidad 1. Si tiene motor, entonces debe tener tren de aterrizaje, y no puede tener pico ni plumas, y es un avión con probabilidad 1. Una vez que se conocen esas variables, no hay posibilidad alguna de cometer un error.

Y eso es lo que hace exactamente nuestro sistema experto, es decir, ha desarrollado una regla tal que no hay posibilidad alguna de que llegue a equivocarse. Lo cual corresponde a sucesos que ocurren con probabilidad 1.

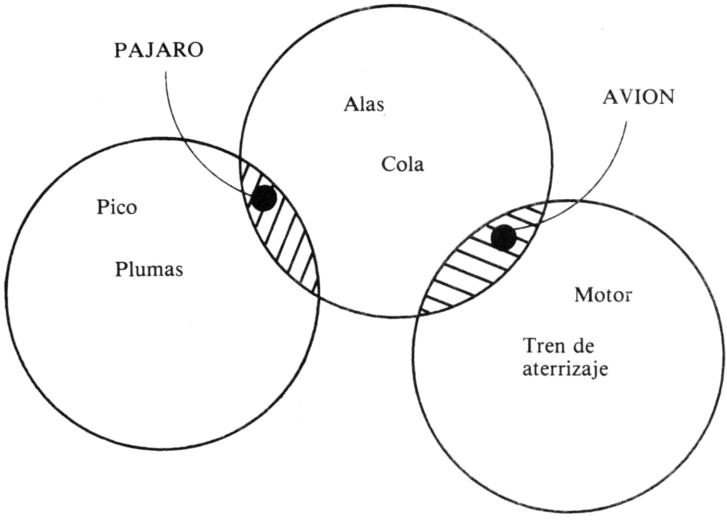


Fig. 3.3
Un pájaro tiene alas, cola, pico y plumas. Un avión tiene alas, cola, motor y tren de aterrizaje.

3.3 OTROS TIPOS DE DATOS

En el Apartado anterior nos hemos concentrado en casos en los que el sistema experto trabaja sobre una simple dicotomía sí/no. Sólo necesita saber si una determinada característica está presente o ausente, y, según sea el caso, la marca con 1 o con 0.

Pero observe que el sistema utiliza 1 y 0, no las palabras Sí y No. Por tanto, puesto que está utilizando números, ¿podría utilizar cualquier número?. La respuesta es Sí. No habría diferencia alguna en su funcionamiento.

Lo cual es una buena noticia. Considere de nuevo el tiempo. Cuando empezábamos, teníamos en cuenta simplemente si hacía frío o no, si llovía o no, si hacía viento o no, si había niebla o no.

Ahora bien, si le echa usted un vistazo al informe meteorológico de ayer, verá que la temperatura (máxima y mínima) puede estar medida en grados, la precipitación en milímetros, la velocidad media del viento en

kilómetros por hora, y la visibilidad en kilómetros (bueno, en Inglaterra, quizás en milímetros). Todas estas cifras nos darían seguramente mucha más información sobre el tiempo que los simples indicadores sí/no. Instintivamente, podría intuirse que estábamos perdiendo algo, si despreciásemos todos estos datos potencialmente útiles.

Por tanto, no los despreciaremos. En lugar de hacer que el experto pregunte si está presente o no una determinada variable, podríamos hacerle imprimir el nombre de cada variable y darle un valor númerico a modo de respuesta. De este modo, podríamos darle la temperatura en grados, la precipitación en milímetros, la visibilidad en kilómetros, y así sucesivamente.

Y si queremos incluir un elemento con respuesta simple sí/no (tal como Truenos), basta con responder 1 para la presencia de esa característica, y 0 para su ausencia.

El programa procederá entonces como antes, pero, afortunadamente, dará mejores resultados, porque le estamos proporcionando mejor información.

Sin embargo, para poder ver con mayor claridad lo que ocurre, en la mayoría de los ejemplos que siguen, utilizaremos datos discretos dicotómicos (Sí/No), pero recuerde que usted podría utilizar variables continuas con la misma facilidad, y que en algunos casos podrían dar mejores resultados.

Por otra parte, no podemos hacer lo mismo con los resultados. El sistema sólo puede escoger entre categorías diferentes de resultado. No puede generar una respuesta que diga cuánta precipitación exactamente va a caer mañana; sólo puede decir si cree o no que va a llover.

Esto no significa que no podamos hacer un intento para conseguir que el sistema sea más concreto. Podríamos decidir, por ejemplo, que la precipitación se clasificase en cinco categorías:

Categorías de precipitación:

Nada	Menos de 10 litros	10-20 l	20-30 l	Más de 30 l
1	2	3	4	5
Seco	Lluvioso	Más lluvioso	Muy lluvioso	Horrible

En tal caso, nuestro sistema experto podría, con cinco resultados, decir algo un poco más preciso sobre el tiempo probable. Hemos visto así cómo puede idearse un sistema con un cierto número de resultados.

Esto no significa que no pueda idearse un sistema que dé respuestas numéricas y no simples categorías. Si lo quisiéramos así, podríamos probar, por ejemplo, a desarrollar una ecuación de regresión múltiple para la precipitación, basada en una gran cantidad de variables de entrada, que daría una estimación exacta (las estimaciones exactas no necesariamente son predicciones correctas) de la precipitación de mañana.

El problema es que, aunque este sistema podría funcionar para predecir la precipitación, no podría modificarse fácilmente para que se hiciese experto en otro campo. Por ejemplo, ¿cómo podría resolver si se trata de un Avión o de un Pájaro? (o, en este caso, un Planeador).

El objetivo de la construcción de su propio sistema experto es conseguir algo que pueda llegar a ser razonablemente experto en muchos campos diferentes. Una especie de dispositivo de propósito general que pueda usted aplicar a cualquier cosa que le interese. Y, siendo así, no tiene sentido hacer que usted pueda predecir el tiempo, si lo que verdaderamente quiere es diagnosticar sus propios síntomas médicos, para poder convertirse en un próspero neurocirujano, u otra cosa por el estilo.

3.4 LA REGLA DE JUICIO

Quizás a estas alturas ya haya puesto en funcionamiento el sistema de aprendizaje, y haya encontrado que, más o menos, funciona. Estupendo, se dice usted. Pero, exactamente. ¿por qué funciona?, ¿qué es lo que hace?, ¿puede conseguirse que haga algo mejor?.

Pues bien, considere la idea del Espacio de Descripción, y no sólo porque es algo útil a considerar, también porque es bastante impresionante poder hablar con conocimiento de causa de otros tipos de espacios diferentes del aburrido tipo tridimensional que tenemos ante nosotros.

Sin embargo, para hacerlo fácil, empezaremos por suponer que está usted sentado ante la mesa de su escritorio, cuyo tablero bidimensional se extiende ante usted, formando un espacio normal y corriente. Y, asimismo, que tiene usted dos conjuntos de objetos, lápices y sujetapapeles, que vamos a utilizar para ilustrar algunos puntos.

Supongamos ahora que quiere separar en dos montones su montón de lápices y sujetapapeles, uno de cada cosa. La tarea es tan fácil que el más tonto puede realizarla: todo lo que hay que hacer es ir tomando los objetos de uno en uno y preguntarse: «¿es un lápiz o un sujetapapeles?», y colocarlo en el montón que le corresponda según la respues-

ta. De hecho, es tan fácil como mirar un objeto y preguntarse si es un pájaro o un avión, e incluirlo en una u otra categoría, dependiendo de lo que sea. Lo importante a destacar es que el problema de la clasificación del objeto es trivial, si la descripción de tal objeto coincide exactamente con los criterios mediante los cuales queremos hacer la clasificación.

Pero supongamos que, por cualquier razón, cuando considera usted el objeto, no puede decir inmediatamente si es un pájaro o un avión, o un lápiz o un sujetapapeles. Entonces, el problema deja definitivamente de ser trivial, y usted tiene que pensar en otro modo de expresar un criterio. Y ésta es exactamente la posición en que se encuentra el ordenador: no puede ver la gran diferencia, aunque para usted sea obvia, por lo que ha de ser un poco más astuto.

Así, tomemos los lápices y los sujetapapeles, y echémoslos sobre la mesa. Coloquemos los lápices a la izquierda y los sujetapapeles a la derecha. Ahora, ignorando el hecho de que puede usted distinguir unos de otros a primera vista, ¿cómo decidiría qué cosa es cada uno?. Pues bien, es fácil. Todo lo que tiene que hacer es trazar una línea por la mitad de la mesa, de modo que todos los lápices caigan a su izquierda y los sujetapapeles a su derecha. Entonces, puede clasificar los dos conjuntos de objetos, sencillamente midiendo su posición en la mesa.

Y eso es lo que, aproximadamente, está haciendo el ordenador. Excepto que él tiene que trabajar con un poco más de dificultad que usted, porque los objetos no se han dejado exactamente en el espacio real. Se han dejado en un espacio de descripción que, en realidad, no es muy diferente.

Volvamos ahora al ejemplo del pájaro/avión, y supongamos disponibles dos elementos de información: ¿vuela el objeto? y ¿tiene plumas el objeto?.

Ahora quite todo de la mesa y escriba a lo largo del tablero «vuela», y, a lo ancho, «plumas», como indica la Figura 3.4. Si un objeto puede volar, se coloca a la derecha de la mesa, y a la izquierda si no puede. Si un objeto tiene plumas se coloca detrás, y si no las tiene, delante.

Obviamente, si tuviese usted un pájaro en la mano(!), lo colocaría atrás y a la derecha. Si tuviese un avión, lo pondría delante y a la derecha. Y si quisiera saber si un objeto es un pájaro o un avión, sin que se le dijera explícitamente lo que es, siempre podría responder con exactitud, mirando simplemente en qué cuadrante de la mesa se encuentra. Podría establecer un juicio, trazando sencillamente una línea entre los dos montones de objetos, y viendo a qué lado de la línea cae cada objeto.

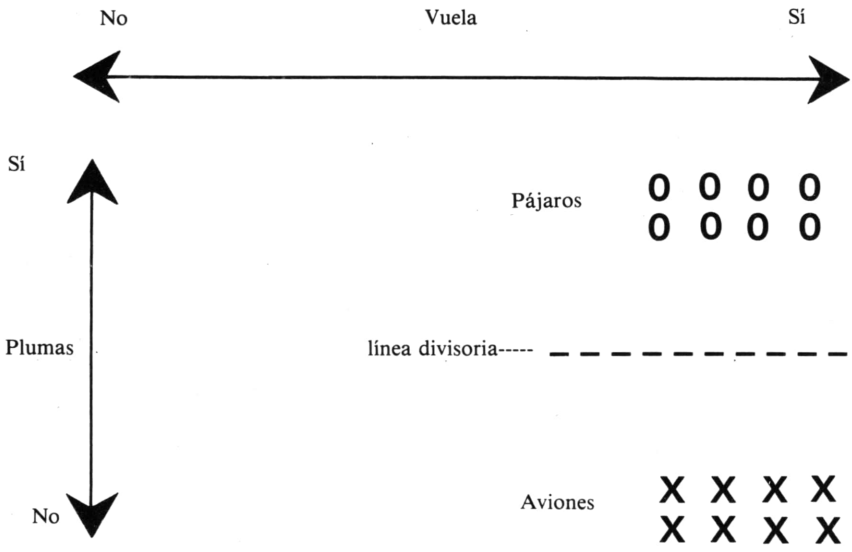


Fig. 3.4
Un Espacio de Descripción que consta de dos Dimensiones denominadas «Vuela» y «Plumas». Los aviones vuelan, pero no tienen plumas. Los pájaros vuelan y tienen plumas.

Y no tienen por qué ser elementos simples Sí/No, como plumas /no plumas. Si quiere, puede considerar, por ejemplo, el pronóstico del tiempo y poner una escala a lo largo del tablero que indique cantidades de precipitación.

Si quita usted la mesa, se queda con los objetos, de cualquier tipo y en cualquier número, suspendidos en el espacio de descripción. Ya no es un espacio definido por la mesa. Y, realmente, tampoco tiene ya nada que ver con el espacio tridimensional normal. Es un nuevo y flamante tipo de espacio, definido únicamente por las variables que usted está utilizando (un eje para cada variable). Y los valores que tienen esas variables para cada objeto definen inequívocamente la posición de ese objeto en el espacio de descripción. Desgraciadamente, no podemos dibujarlo para que se vea mejor.

Para ilustrar la idea en tres dimensiones, suponga que está usted paseando en su jardín en una agradable tarde de verano (a veces, en Inglaterra tenemos tardes de verano agradables). Suponga que ve usted dos nubes de mosquitos que zumban bajo el sol. Una de las nubes está for-

mada por Grandes Mosquitos, y la otra por Pequeños Mosquitos, y se le ocurre la travesura de separar una de otra. Un método que podría servir, sería conseguir la hoja de papel más grande del mundo y colocarla entre las dos nubes de mosquitos.

Es cierto que esto podría servir, pero, antes de que lo intente, hemos de decir que la única razón para dar este ejemplo es mostrar que usted puede tener dos grupos de objetos definidos con relación a tres dimensiones (la posición física de cada mosquito), y que, en teoría, puede separarlos mediante la colocación de una superficie entre ellos.

De hecho, una vez que se ha comprendido lo que es el espacio de descripción, pueden definirse los objetos sobre cualquier número de dimensiones. Si tenemos una dimensión para cada variable medida, un conjunto de objetos definidos en términos de diez variables existe en un espacio de descripción de diez dimensiones. Y la idea de colocar una superficie entre ellos es exactamente la misma: podemos tener una superficie de diez dimensiones para nuestras diez variables, y clasificar cada objeto según a qué lado de la superficie se encuentra.

A propósito, la ecuación de una superficie en n dimensiones es $y = b_1x_1b_2x_2 + \ldots + b_nx_n$, donde x_i representan medidas de «posición» sobre los n ejes, y b_i son constantes. Así, el tablero de la mesa tiene la ecuación, en dos dimensiones, $y = b_1x_1 + b_2x_2$.(*)

(*)N. del T. Creemos que aquí se hace necesaria una precisión: dado un plano (como la superficie de la mesa), definidos en él unos ejes coordenados, x_1 y x_2, y fijadas unas constantes, b_1 y b_2, lo que realmente representa la ecuación $y = b_1x_1 + b_2x_2$ es, para cada valor de y, una recta dentro de ese plano (o espacio bidimensional). Así, para $y = 0$ se trata de una recta que pasa por el origen ($x_1 = x_2 = 0$), con pendiente igual a b_1/b_2, y, para cualquier otro valor de y, es una recta paralela a la anterior. Análogamente, en un espacio tridimensional, $y = b_1x_1 + b_2x_2 + b_3x_3$ representa una familia de planos paralelos (uno para cada valor de y). Y en un espacio de n dimensiones, para cada valor de y, $y = b_1x_1 + b_2x_2 + \ldots + b_nx_n$ será la ecuación de un plano en ese espacio (aunque el autor utiliza la palabra «superficie», al ser la ecuación lineal, no es una superficie cualquiera, sino plana).

La superficie (plana) vendrá entonces determinada tanto por los valores de b_i como por el de y. Dadas las coordenadas x_i de un objeto (considerado éste como un punto en el espacio), se puede calcular el valor resultante para y, que, si es igual al de la superficie, indicará que el objeto está sobre esa misma superficie, y si es mayor o menor, que está a un lado u otro de la misma. Por ejemplo, en la Figura 3.4, la «superficie» es la «línea divisoria», con ecuación $x_2 = 0$. Si un objeto tiene $x_2 > 0$, es un pájaro, y si tiene $x_2 < 0$, es un avión. En este caso, la dimensión x_1 («vuela») no influye para la clasificación, lo que se refleja, en la ecuación de la «superficie», en que no figura la variable x_1.

Especificando los valores de x_i para un objeto nuevo, podemos calcular el valor de y para ese objeto, y podemos usar ese valor de y para decir a qué lado de la superficie se encuentra el objeto.

Y esto es lo que hace el programa de aprendizaje. Toma todos los x_i que le damos a través de ejemplos, y calcula valores para b_i, de modo que, una vez que el sistema funciona correctamente, obtiene un valor de y que determina inequívocamente a qué grupo pertenece el objeto.

Puede usted considerarlo como un proceso de prueba y error. Usted le presenta al programa un objeto mediante valores de las variables. Inicialmente, el programa tiene una superficie que está situada en cualquier parte, mira a qué lado de la superficie cae el objeto, y decide de acuerdo a ello. Por casualidad, puede acertar. Pero, si se equivoca, cambia ligeramente la posición de la superficie en el espacio de descripción, utilizando los valores que le ha dado usted. Hace luego otra tentativa con un nuevo objeto, quizás moviendo un poco más la superficie, y luego otra, moviéndola un poco más... Y así sucesivamente, hasta que ha posicionado la superficie de tal modo que siempre da las respuestas correctas (si es que tal cosa es posible).

Después de esta explicación, puede que se pregunte usted si hay un forma «óptima» de utilizar este método. Por ejemplo, ¿no sería mejor ajustar de algún modo la posición de la superficie no sólo cuando el ordenador se equivoca, sino también cuando acierta?.

La respuesta es sencilla: todo depende del tipo de problema que le dé usted al ordenador; pero el encontrar la forma óptima no es tan sencillo. Si tiene algunos datos a presentar a su sistema experto, vale la pena que tenga en mente las siguientes observaciones:

Si sólo hay un número finito de ejemplos posibles que puedan darse, obtendrá los mejores resultados dándole al experto todos esos posibles valores para que aprenda de ellos.

Puede que el llevar la superficie (la regla de juicio) hasta la posición correcta requiera algún tiempo, porque a menudo no basta con dar una vez cada ejemplo; puede que tenga que presentar cada ejemplo cierto número de veces.

Es difícil decir de antemano cuántos ensayos va a necesitar exactamente el sistema para llegar a funcionar correctamente; por tanto, deberá usted trabajar pragmáticamente (lo que significa probar y observar). Déle al experto un conjunto de ejemplos, déjele probar suerte y cuente cuántos errores comete. Déjele luego probar de nuevo, y vuelva a contar

los errores. Continúe así hasta que llegue a no cometer ningún error, o hasta que ya no mejore más.

Si quiere usted que trabaje con objetos cuya clasificación todavía no conoce (como, por ejemplo, el pronóstico del tiempo para alguna fecha futura), procure comprobar que los ejemplos que le da para que aprenda son lo más parecidos posible a los objetos con los que eventualmente tendrá que enfrentarse. Esto puede parecer obvio, pero observe que, si usted lo entrena en una cosa, puede que no necesariamente sea capaz de utilizar las mismas reglas con otros datos diferentes.

Hay un punto sobre el que quizás esté usted preguntándose: si los sistemas expertos sirven para sustituir a los expertos humanos mediante la incorporación de experiencia humana, ¿cómo es que el sistema elabora por sí mismo un conjunto de reglas?. Casi parece como si no necesitara ayuda humana en absoluto. La respuesta es que, quizás sin darse cuenta, usted le ha proporcionado bastante conocimiento muy humano. Lo que le ha dado es el conocimiento inicial de qué era cada uno de los objetos; si el programa ha podido aprender algo, sólo ha sido gracias a que usted, el usuario, era capaz de juzgar la diferencia entre un pájaro y un avión.

3.5 CÓMO CONSTRUIR UNA REGLA

Una vez que se ha percatado usted de que todo lo que hace el sistema de aprendizaje es encontrar, en un espacio de descripción de n dimensiones, la ecuación de una superficie que separa razonablemente bien dos grupos de objetos, puede que piense en la posibilidad de hacer algo un poco mejor.

Después de todo, el sistema de aprendizaje encuentra la superficie siguiendo un procedimiento aparentemente tosco y aproximado, y a menudo necesita un aprendizaje bastante largo. Por tanto, ¿por qué no calcular un conjunto de valores realmente bueno para una superficie, y dárselos al sistema experto desde el principio?. De ese modo, no sería necesario entrenamiento alguno, y, en los casos en que resulta difícil distinguir entre objetos, al menos sabríamos que estamos utilizando la mejor superficie posible.

Pues bien, aunque no quisiéramos desanimar a nadie, debemos decir que el asunto de la discriminación es bastante difícil. A propósito, unos hablan de «discriminación» y otros de «clasificación», pero todos se refieren a lo mismo. Y en los libros sobre análisis multivariable (multivariable = gran cantidad de variables) es típico encontrar uno o dos capítulos sobre el problema.

Considere de nuevo el problema de las dos nubes de mosquitos que usted quería separar. Todo irá bien siempre y cuando las dos nubes estén razonablemente alejadas una de otra. Pero si no lo están, ya no sólo el sistema de aprendizaje, sino casi cualquier sistema va a cometer errores, porque no hay posición posible en la que se pueda colocar una superficie que separe los dos grupos.

Pero lo que sí podría hacerse sería calcular exactamente el centro de cada grupo, trazar una línea entre los dos centros y clasificar cada elemento según que caiga a un lado y otro del punto central de la línea. Esta es, de hecho, la idea básica de la mayoría de los métodos de discriminación. Por muchos motivos, es mejor proceder así que confiar en el sistema de aprendizaje, porque éste depende mucho de los valores alejados cuando decide la posición final de la superficie. Es decir, sólo altera la superficie cuando comete errores, y, al final del proceso, sólo comete errores cuando, en cualquier caso, está cerca de la posición correcta. Así, si piensa usted en las dos nubes de mosquitos, la superficie se mueve a un lado y otro por el capricho de unos pocos mosquitos que se encuentran en las afueras de sus nubes, y estos pocos podrían no ser muy representativos del conjunto de mosquitos. Y lo mismo ocurre con los objetos en el espacio de descripción. Los objetos alejados pueden no ser representativos, y conducir a resultados incorrectos.

La situación en que nos encontramos es la de un problema de clasificación de objetos en un espacio de N dimensiones. Por tanto, comencemos por reservar espacio,

```
10  INPUT «M1 = »;M1: INPUT «M2 = »;M2: INPUT «N = »;N
20  DIM V1(N), V2(N)
```

para las N variables y luego busquemos M ejemplos. Supongamos que tenemos M1 del primer tipo, y M2 del segundo

```
30    FOR I = 1 TO N
40    V1(I) = 0
50    NEXT
60    FOR I = 1 TO M1
70    FOR J = 1 TO N
80    INPUT «VARIABLE = »;X
90    V1(J) = V1(J) + X/M1
100   NEXT:NEXT
```

donde X es el valor de la variable J para el ejemplo I.

Haga lo mismo para los V2(J) de los M2 ejemplos del segundo tipo,

y tendrá así los valores medios o «medias» de cada uno de los dos tipos de los vectores V1 y V2:

```
110    FOR I=1 TO N
120    V2(I)=0
130    NEXT
140    FOR I=1 TO M2
150    FOR J=1 TO N
160    INPUT «VARIABLE=»;X
170    V2(J)=V2(J)+X/M2
180    NEXT:NEXT
```

Ahora, trate de clasificar otro objeto, utilizando esas medias. Suponga que introduce los valores de las variables de ese objeto en el vector X. No olvide DIM X(N).

Las «distancias», D1 y D2 (*), del nuevo objeto a las dos medias pueden calcularse así:

```
190    DIMX(N)
200    D1=0: D2=0
210    FOR J=1 TO N
220    INPUT X(J)
230    D1=D1+X(J)*V1(J)
240    D2=D2+X(J)*V2(J)
250    NEXT
```

Otra posibilidad es calcular la diferencia entre las medias y guardarla en un nuevo vector, V:

```
260    DIM V(N)
270    FOR J=1 TO N: V(J)=0: NEXT
280    FOR J=1 TO N
290    V(J)=V1(J)-V2(J)
300    NEXT
```

Y luego calcular D a partir de los vectores V y X, y clasificar el objeto según que D sea mayor o menor que cero:

(*) N. del T. El lector que tenga unos conocimientos mínimos de geometría, se percatará de que D1 y D2, tal como se calculan en las líneas 230 y 240 de programa, no representan en absoluto «distancias». De hecho, cualquiera puede comprobar, programándolo, que el algoritmo que se sugiere en el resto de este Apartado no funciona (salvo en casos muy especiales, uno de los cuales es el del pájaro y el avión). El cálculo correcto de las distancias puede hacerse como se indica en este mismo libro, en el Apartado 6.3.

```
310    D=0
320    FOR I=1 TO N
330    D=D+X(I)*V(I)
340    NEXT
```

En este momento debe resultar bastante evidente que todo esto, en términos de programa, es muy similar al método utilizado por el sistema de aprendizaje.

De hecho, es posible modificar el sistema de aprendizaje para que se transforme en el método de la «media más cercana»:

En lugar de ajustar la regla de decisión sólo cuando se comete un error, dele un ejemplo al sistema, y dígale a qué clase pertenece. Modifique entonces la regla de decisión, de modo que contenga siempre el último valor de la diferencia entre los dos conjuntos de valores medios.

Por ejemplo, si hasta ahora ha habido M1 ejemplos de la clase 1 y añade usted otro,

```
370    FOR J=1 TO N
380    V1(J)=(V1(J)*M1+X(J))/(M1+1)
390    NEXT
```

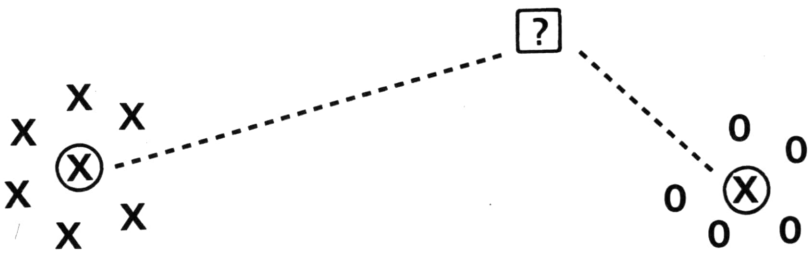


Fig. 3.5
Medida de la distancia a la media más cercana

donde el vector X contiene los valores correspondientes al nuevo objeto.

De esta forma, V1 se actualiza constantemente con la última estimación del valor medio de ese grupo.

Y, utilizando los valores más recientes de V1 y V2, es posible tener los valores más recientes de la regla de decisión:

```
FOR J=1 TO N: R(J)=0: NEXT
FOR J=1 TO N
R(J)=V1(J)-V2(J)
NEXT
```

y tomar luego la decisión pertinente, según que D sea mayor o menor que cero, calculando D así:

```
D=0
FOR J=1 TO N
D=D+X(J)*R(J)
NEXT
```

Pueden utilizarse muchos otros métodos para construir reglas de decisión, pero todos ellos dependen de un conocimiento más profundo de las reglas que se usan. Entretanto, si quiere usted saber cuál de los dos métodos es mejor —el algoritmo de aprendizaje o el enfoque de la media más cercana— lo mejor es ensayarlos y observar. No hay una respuesta general, porque depende mucho de la naturaleza exacta de las variables que personalmente elija usted para darle al sistema.

3.6 PROBABILIDADES A PRIORI

Pero hay un perfeccionamiento del enfoque de la media más cercana que puede usted probar. Ocurre en el caso en que sabe usted con seguridad que un resultado tiene más posibilidades que otro, independientemente de los valores de las variables. Esto es lo que se conoce como probabilidad a priori de cada resultado: si el resultado 1 tiene tres veces más posibilidades de ocurrir que el resultado 2, entonces P(resultado 1) = 0,75, y P(resultado 2) = 0,25. Lo que puede usted hacer en este caso es tomar una decisión que no esté totalmente basada en la media más cercana.

Considere lo siguiente: para tomar una decisión basada en la media más cercana, estando la diferencia entre las dos medias contenida en el vector R, simplemente se considera si el valor calculado para D es mayor o menor que cero. Pero si, para algún ejemplo, resultara D próximo a cero, y usted supiera que la clase 1, por ejemplo, es más probable que la clase 2, podría decidirse por la clase 1, aún cuando el valor calculado para D tienda a sugerir lo opuesto.

En términos de programación, ello equivale, simplemente, a comprobar si D es mayor que un determinado valor C. Anteriormente

hemos tomado C = 0, y la cuestión es que, en ciertos casos, el sistema podría mejorar con algún otro valor de C.

La dificultad radica en especificar un valor para C realmente bueno y distinto de cero. Si usted sabe lo suficiente sobre los datos con los que está trabajando, y es un matemático competente, es posible que encuentre un valor que le permita decir cosas bastante concretas sobre cómo se comportará el sistema. Pero, si no es así, entonces hay que volver a experimentar (lo que, en cualquier caso, es una buena idea), y la única advertencia importante que hay que hacer es que si no tiene usted suficientes ejemplos típicos para comprobar el sistema, entonces jamás llegará realmente a saber cómo va a funcionar en la práctica, y lo mejor es evitarse complicaciones.

3.7 CÓMO TENER MÁS OPCIONES

Imagínese a sí mismo mirando al cielo. Todos a su alrededor están gritando: «¿será un pájaro?, ¿será un avión?». Saca usted su sistema experto, teclea unas cuantas variables bien elegidas, y, en cuestión de segundos; con toda autoridad, se pronuncia sobre el asunto. «Falso —exclaman todos con tono de regocijo malévolo—. Es un planeador».

Y con las predicciones meteorológicas la cosa no es mejor. Volviendo a nuestro cuidadosamente elaborado sistema experto, resulta que puede predecir un día lluvioso o un día seco. Pero, en ambos casos, si al día siguiente se encuentra usted caminando a tientas en medio de una niebla impenetrable, seguramente se sentirá decepcionado. La niebla, piensa usted, es una especie de lluvia, pero no es exactamente lluvia. Y también es como ausencia de lluvia. Lo que la niebla es, reflexiona, es niebla. Y sólo eso.

Y, en cierto modo, se sentirá defraudado de que su sistema experto no haya podido prever otros resultados.

En general, lo que usted quiere es que su sistema experto pueda dar cualquier número de resultados diferentes. Podría conformarse si tuviera que especificar previamente esos resultados, pero, en cualquier caso, quiere tener más de dos resultados posibles.

Por supuesto, hay muchas formas de construir un sistema experto que pueda dar una variedad de resultados. Pero, por el momento, nos limitaremos a considerar una extensión del sistema con el que hemos trabajado hasta aquí.

Recuerde que teníamos una regla para tomar decisiones. Era una sola regla, y el sistema experto toma una decisión —una decisión del tipo «si o no»— sobre la base de esa única regla. Lo que vamos a hacer ahora es dotar al sistema de un número cualquiera de reglas, de modo que pueda pronunciarse sobre un número cualquiera de resultados posibles. Para ello, vamos a preparar una matriz bidimensional en la que guardar las reglas; esta matriz va a parecerse mucho a la matriz rectangular en la que poníamos las probabilidades.

Llamaremos a esa matriz R(V,Q): reglas para V variables con Q resultados.

Y utilizaremos de nuevo el ejemplo del pájaro/avión:

	Pájaro	Avión
Alas	0	0
Cola	0	0
Pico	0	0
Motor	0	0
Plumas	0	0
Tren de aterrizaje	0	0

De momento, nos olvidaremos de los planeadores y, con 6 variables, tenemos R(6,2), es decir, Q = 2 resultados.

El método de trabajo es el siguiente:

Cada columna de la matriz R contiene una regla que se utiliza para indicar la confianza que el sistema experto tiene en el resultado correspondiente a esa columna. La magnitud de esa confianza se mide mediante el cálculo de D, que, como antes, se hace utilizando el vector V, que contiene los valores de las variables que se consideran en cada momento.

Es decir, para el resultado J (J = 1 ó 2) calculamos D así:
D = D + R(I,J)*V(I) desde I = 1 hasta I = 6

Observe que todos los R(I,J) se inicializan con el valor cero, por lo que también todos los D serán nulos inicialmente. En otras palabras, la confianza (o falta de confianza) inicial es la misma para los dos resultados.

Ahora, dependiendo de cómo quiera usted que la cosa funcione, o bien le dice al experto a qué grupo pertenece el vector V, o le deja averiguarlo y, si se equivoca, hace que actúe así:

Si V pertenece al resultado Pájaro, entonces sumamos los valores de V(I) a la columna de Pájaro, y los restamos de la columna Avión. Si V perteneciese a Avión, sumaríamos los valores de V(I) a Avión, y los restaríamos de Pájaro. Esto es lo que resulta despues de sumar los valores de «Pájaro»:

	V(I) Pájaro	V(I) Avión	R(I,1) Pájaro	R(I,2) Avión
Alas	1	1	1	—1
Cola	1	1	1	—1
Pico	1	0	1	—1
Motor	0	1	0	0
Plumas	1	0	1	—1
Tren de aterrizaje	0	1	0	0

Hasta ahora, hemos presentado a la matriz R un V de Pájaro. El sistema no ha podido tomar ninguna decisión, porque todos los D eran iguales. Por tanto, ha sumado V(I) a la primera columna, y lo ha restado de la segunda. Si ahora recibe de nuevo «Pájaro», puede tomar una decisión correcta, porque D resulta +4 usando la columna R(I,1) y —4 usando R(I,2): todo lo que ha de hacer el experto es seleccionar la columna con la que se obtenga mayor valor para D.

Pero si ahora le damos al experto un V(I) correspondiente a Avión, nos encontramos con que sigue seleccionando «Pájaro» como resultado más verosímil, porque con R(I,1) se obtiene para D un valor de +2, y con R(I,2) se obtiene —2.

Por tanto, nuestro maravilloso sistema experto todavía no puede siquiera distinguir un pájaro de un avión.

Pero sumemos el V(I) de Avión a la segunda columna, y restémoslo de la primera, y obtendremos:

	R(I,1) Pájaro	R(1,2) Avión
Alas	0	0
Cola	0	0
Pico	1	—1
Motor	—1	1
Plumas	1	—1
Tren de aterrizaje	—1	1

Si ahora damos los valores de V(I) correspondientes a Avión, la primera columna da para D un valor de —2, y la segunda, +2. Con lo cual, el sistema elegirá el resultado Avión. Correcto.

Y si volvemos a comprobar con el V(I) de Pájaro, vemos que da +2 para la columna primera, y —2 para la segunda.

Por tanto, el sistema experto funciona ahora tan bien como lo hacía antes. De hecho, si repasa usted lo anterior, verá que el sistema está haciendo exactamente lo mismo que cuando sólo tenía un vector («array» unidimensional) de regla, por lo que puede que piense usted, quizá con cierto cinismo, que todo este esfuerzo no ha servido para nada.

Y si sólo se tratara de distinguir entre un pájaro y un avión, realmente habría sido un esfuerzo baldío.

Pero volvamos a considerar aquel planeador, y añadamos otra columna a R(I,J) para él. Procederemos exactamente como antes: calculando los valores de D para cada columna, y eligiendo la columna con la que se obtiene el mayor valor de D. En caso de error, o de empate, le decimos al sistema qué columna debería haber elegido. Entonces, el sistema sumará los valores de V(I) con los que está trabajando a la columna correcta, y los restará de todas las otras columnas que hubieran dado un valor de D mayor o igual al de la columna que debería haber elegido.

Si prefiere decirlo así, el sistema promociona la regla del resultado correcto, y rebaja los valores de las reglas de los otros resultados, para alejarlas de la correcta.

Supongamos que tenemos un planeador. Un planeador es como un avión, pero no tiene motor ni tren de aterrizaje, claro.

En el primer ensayo, todos los R(I,J) son nulos, de modo que podría elegir igual Planeador que los otros dos resultados. Por tanto, si damos Pájaro, se rebajarán la tercera columna (Planeador) y la segunda (Avión).

Si en el segundo ensayo damos Avión, el sistema da como resultado Pájaro. Por tanto, sumamos el V(I) de avión a la segunda columna y lo restamos de la primera. Pero también hemos de restarlo de la tercera columna (Planeador), que hasta ahora era igual a la de Avión. Con ello resulta:

	R(I,1) Pájaro	R(I,2) Avión	R(I,3) Planeador
Alas	0	0	—2
Cola	0	0	—2
Pico	1	—1	—1
Motor	—1	1	—1
Plumas	1	—1	—1
Tren de aterrizaje	—1	1	—1

Evidentemente, con estos valores el sistema nunca eligiría Planeador, porque como todos los valores de la columna tercera son negativos, alguna de las otras reglas dará siempre un valor mayor para D. Pero, en cualquier caso, démosle los valores de V(I) de Planeador:

	V(I) Planeador	R(I,3) Planeador
Alas	1	—1
Cola	1	—1
Pico	0	—1
Motor	0	—1
Plumas	0	—1
Tren de aterrizaje	0	—1

No necesitamos escribir ya las otras dos columnas, porque cada vez que damos el V(I) de Planeador y el experto no acierta, esos valores de V(I) se restan de las columnas que corresponden a Pájaro y a Avión: se resta una unidad de alas y de cola. Ello no va a afectar en absoluto a la capacidad del sistema para distinguir entre pájaro y avión. Pero, al ir sumando una unidad a alas y cola en la columna de planeador, esos números irán incrementándose hasta llegar a unos valores con los que, para V(I) de Planeador, R(I,3) dará el máximo valor de D, y la elección ya será correcta en lo sucesivo.

En ese momento, R(I,J) tendrá este aspecto:

	R(I,1) Pájaro	R(I,2) Avión	R(I,3) Planeador
Alas	—1	—1	0
Cola	—1	—1	0
Pico	1	—1	—1
Motor	—1	1	—1
Plumas	1	—1	—1
Tren de aterrizaje	—1	1	—1

Usted mismo puede comprobar que, a partir de ahora, el sistema elegirá ya sin problemas entre las tres alternativas. Ha aprendido correctamente y ya no comete más errores.

Y, lo que es más importante, podría tener *millones* de resultados diferentes y *millones* de variables diferentes, y el sistema siempre iría manipulando los números hasta encontrar un conjunto de reglas correctas para escoger un resultado a partir de un determinado conjunto de variables.

3.8 ¿PUEDE EQUIVOCARSE EL SISTEMA?

Hasta aquí, bien. Pero ¿qué posibilidades hay de que la cosa vaya mal? (es decir, de que el sistema no llegue nunca a aprender a distinguir correctamente entre los resultados).

Pues bien, es bastante difícil determinar esas posibilidades en términos de probabilidades exactas, porque todo depende mucho del tipo de problema que le plantee usted al experto. Todo el método depende del concepto de Separabilidad Lineal, y ésta es una buena frase para recordar.

—«Naturalmente —dice usted, apoyándose despreocupadamente sobre la barra del bar—, mi Sistema Experto, una vez que ha construido su Base de Conocimientos, funciona perfectamente». En este momento usted puede hacer la prueba, para impresionar, de darle una pensativa chupada a su pipa y agregar: «Siempre y cuando trabaje sobre un problema que sea Linealmente Separable, desde luego». Y luego puede soltar una carcajada y, con suficiencia, exclamar: «¡Pero decidme de uno que no lo sea!».

Esta impresionante jerigonza intimidará hasta a sus más severos críticos, y disimulará fácilmente el hecho de que su sistema experto no siempre funciona.

No tenemos más que volver a nuestras nubes de mosquitos. El sistema experto aprenderá a distinguirlas, siempre y cuando se pueda trazar una línea o colocar una superficie entre los dos grupos. Si no puede, el sistema no aprenderá. Así de fácil.

Considere, sin embargo, el ejemplo de los postes rojos y azules de una empalizada de jardín.

Usted tiene un jardín con una cerca de postes a lo largo de un lateral. Los postes van alternándose: uno rojo, uno azul, uno rojo, uno azul (no es demasiado difícil de entender). ¿Son linealmente separables estos postes?. ¿Puede usted distinguir entre los rojos y los azules, colocando una superficie entre ellos?.

Pues bien, depende. Si usted los mide (o los describe) en función de la distancia a lo largo del jardín, entonces no puede. Supongamos que los ha descrito así. A tres metros del comienzo, hay un poste rojo. Medio metro más allá hay otro, y es azul. Coloca usted una superficie a una distancia de 3,25 metros entre los dos postes. Evidentemente, tal superficie permite distinguir entre esos dos postes (uno rojo y uno azul), pero no entre cualquiera de los que quedan a un lado y otro de la superficie.

Los postes, tal como los ha descrito usted, no son linealmente separables.

Sin embargo, si baja usted al jardín y desplaza un poco hacia la derecha cada uno de los postes rojos, y un poco hacia la izquierda cada uno de los azules, entonces puede colocar fácilmente una superficie entre ellos: basta ponerla a lo largo de la hilera original de postes. Siguen siendo los mismos postes, pero ahora son linealmente separables.

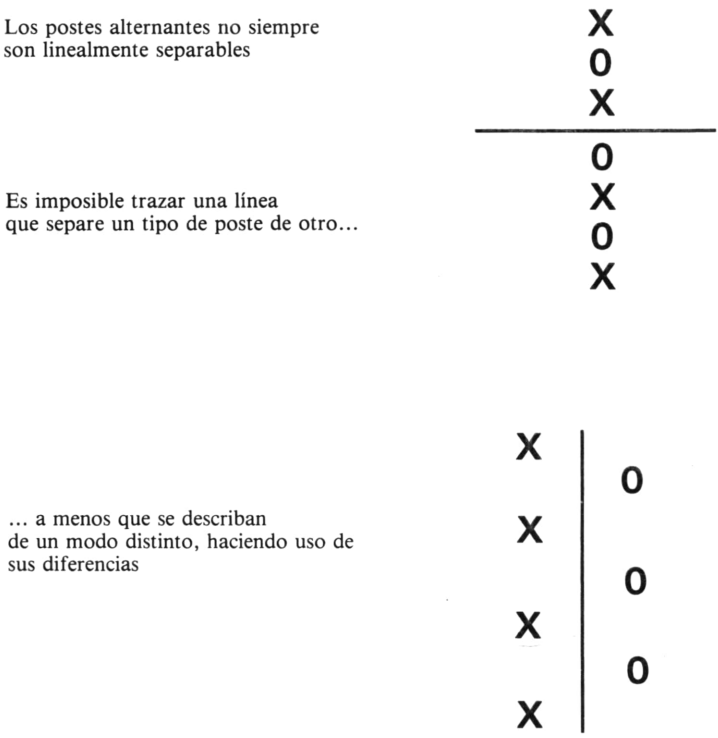


Fig. 3.6
El problema de los postes alternantes

Otra posibilidad consiste en no describirlos en función de donde se encuentran en el jardín, sino simplemente numerarlos por orden. Como se van alternando, es evidente que los postes a los que correspondan nú-

meros impares serán de un color, y los pares de otro. Por tanto, si usted describe de este modo a los postes, serán linealmente separables, y nuestro sistema experto podrá aprender a distinguir entre los azules y los rojos.

Lo importante es que los postes no han cambiado, no ha tenido usted que moverlos. Sencillamente, los ha descrito de otro modo.

Esta es la esencia del asunto. Usted debe tratar de elegir aquellas variables que parezca que van a permitir al sistema experto distinguir entre los distintos resultados. Para ser francos, no es ésta una observación muy aguda, porque en cualquier caso usted tenderá a hacerlo así de una manera natural. Después de todo, si quiere usted que su sistema experto le diga si va a llover mañana o no, no es verosímil que le dé como datos los resultados de fútbol, por ejemplo. El conocimiento de los resultados de los partidos de hoy no va ayudar a predecir el tiempo que hará mañana, por lo que, si usted diera esos datos, no estaría permitiendo que el problema fuese linealmente separable.

(Podría admitirse que los resultados de los partidos pueden decir algo sobre el tiempo de hoy, de donde podría deducirse algo sobre el tiempo de mañana, pero difícilmente puede uno esperar de ello unas respuestas fiables).

Aparte de la naturaleza del propio problema (¿son los casos linealmente separables?) está la cuestión de cuánto tiempo va a tardar el sistema experto en aprender un conjunto de reglas que sea aceptable.

Lo mejor, si el número de posibilidades es relativamente pequeño, es darle todos los ejemplos posibles hasta que deje de cometer errores. Si hay un número enorme de posiblidades, entonces hay que darle al menos un par de ejemplos para cada unos de los resultados posibles, y vigilar su funcionamiento hasta que parezca haberse estabilizado.

3.9 RECAPITULACIÓN: EL PROGRAMA ACTUALIZADO

El programa que damos a continuación funciona utilizando los métodos que hemos descrito hasta aquí, pero vale la pena repasarlo para asegurarse de que no contiene errores.

En primer lugar, pregunta cuántas variables se van a considerar en el proceso de toma de decisión y cuántos resultados puede haber; estos números son los valores de V y Q. Conociendo esos valores, se pueden DI-

Mensionar los «arrays» (vectores y matrices): V(V) para guardar los valores de las variables del ejemplo en curso; V$(V) para guardar los nombres de las variables; Q$(Q) para guardar los nombres de los resultados; R(V,Q) para guardar las reglas de decisión de las V variables para los Q resultados posibles; D(Q) para guardar los valores calculados correspondientes a un determinado V(I), utilizando R(I,J) para los Q resultados posibles.

Se introducen los nombres de las variables y de los resultados para que puedan mencionarse mediante esos nombres.

En la línea 120 comienza una sesión de entrenamiento. Se introducen valores para V(I), y se calculan los Q valores de D(J) utilizando R(I,J).

El sistema busca el mayor valor en el vector D, y supone que este mayor valor, D(J) = D(MY), es el que indica el resultado correcto, Q$(MY).

Si éste es el resultado correcto, el programa vuelve a la línea 120 para continuar con otro ejemplo. Si no lo es, todos los posibles resultados se presentan en la pantalla, y el programa pregunta cuál era el correcto. Este será Q$(MJ).

Utilizando este conocimiento, el sistema reajusta R(I,J) adecuadamente, restando V(I) de cada regla que hubiera dado un valor igual o mayor que D(MJ), y sumando V(I) a la regla MJ, es decir, a R(I,MJ).

Una vez ajustada R(I,J), el programa vuelve otra vez a la línea 120 para considerar otro ejemplo.

Conforme avanza la sesión de entrenamiento, el juicio del sistema debe ir mejorando, y, en ciertos casos, como en el ejemplo del pájaro/avión/planeador, llegará a acertar siempre.

El programa también incorpora una pequeña evaluación, que le permite comprobar por sí mismo cuándo ha llegado a aprender perfectamente.

Listado para el Apple II:

```
10  HOME : INPUT "CUANTAS VARIABLES
    HAY? ";V
20  DIM V(V),V$(V)
30  PRINT "INTRODUZCA LOS NOMBRES DE
     LAS VARIABLES"
```

```
40  FOR I = 1 TO V
50  PRINT "LA VARIABLE ";I;" ES ";: INPUT
    V$(I)
60  NEXT I
70  INPUT "CUANTOS RESULTADOS HAY? "
    ;Q: DIM Q$(Q),R(V,Q),D(Q),E(Q): REM
     E(Q) SIRVE PARA HACER UNA EVALU
    ACION
80  PRINT "INTRODUZCA LOS NOMBRES DE
     LOS RESULTADOS"
90  FOR I = 1 TO Q:E(I) = 0
100  PRINT "EL RESULTADO ";I;" ES ";
    : INPUT Q$(I)
110  NEXT I
120  HOME : PRINT "ENTRO EN UNA SESI
    ON DE ENTRENAMIENTO": PRINT "DEM
    E VALORES DE LAS VARIABLES": PRINT
    "Y YO LE DARE UN RESULTADO": PRINT
    "USTED ME DIRA SI ES CORRECTO O
    NO"
130 D = 0: FOR I = 1 TO Q:D(I) = 0: NEXT
    I
140  FOR I = 1 TO V
150  PRINT "LA VARIABLE ";I;" (";V$(
    I);") VALE ";: INPUT V(I)
160  NEXT I
170  FOR I = 1 TO V
180  FOR J = 1 TO Q
190 D(J) = D(J) + V(I) * R(I,J)
200  NEXT J: NEXT I
210  FOR I = 1 TO Q
220  IF D(I) >  = D THEN D = D(I):MY
     = I
230  NEXT I
240  PRINT "EL RESULTADO ES ";MY;" (
    ";Q$(MY);")? "
250  INPUT R$
260  IF R$ = "S" THEN E(MY) = 1:EV =
    0: FOR I = 1 TO Q:EV = EV + E(I)
    : NEXT I: IF EV = Q THEN  PRINT
    "SOY PERFECTO!": END
```

```
270  IF R$ = "S" THEN  GOTO 120
280  FOR I = 1 TO Q: PRINT I;" ";Q$(
     I)
290  NEXT I
300  INPUT "CUAL ES EL RESULTADO? ";
     MJ
310  FOR I = 1 TO Q
320  IF D(I) >  = D AND I <  > MJ THEN
     FOR J = 1 TO V:R(J,I) = R(J,I) -
     V(J): NEXT J
330  NEXT I
340  FOR J = 1 TO V
350 R(J,MJ) = R(J,MJ) + V(J)
360  NEXT J
370  PRINT "ANTES DE ESTE ERROR HABI
     A CONSEGUIDO ";EV;" ACIERTOS": PRINT
     : PRINT "PULSE CUALQUIER TECLA P
     ARA SEGUIR": GET X$
380  FOR I = 1 TO Q:E(I) = 0: NEXT I

390  GOTO 120
```

Listado para el Sinclair Spectrum:

```
  10 CLS
  20 INPUT "Cuantas variables? ";v
  25 DIM v(v): DIM v$(v,18)
  30 PRINT AT 16,0;"Por favor, nombralas:"
  40 FOR i=1 TO v: INPUT "La variable ";(i
);" es ";v$(i): NEXT i
  50 CLS
  70 INPUT "Cuantos resultados tienes? ";q
  75 DIM q$(q,10): DIM r(v,q): DIM d(q): D
IM s(q)
  80 PRINT AT 16,0;"Por favor, nombralos"
  90 FOR i=1 TO q: LET s(i)=0: INPUT "El r
esultado ";(i);" es ";q$(i): NEXT i
 100 CLS
 110 LET sc=0
 120 PRINT "Esta es una sesion de entrenam
iento": PRINT : PRINT "Por favor, da valor
es a las ": PRINT "variables": PRINT : PRI
```

```
NT "Aproximare un resultado": PRINT : PRIN
T "Dime si es correcto"
 125 PRINT : PRINT "Pulsa cualquier tecla
para continuar"
 126 LET a$=INKEY$: IF a$="" THEN  GO TO 1
26
 128 CLS : PRINT AT 16,0;"Inroduce las var
iables:"
 130 LET d=0: FOR i=1 TO q: LET d(i)=0: NE
XT i
 140 FOR i=1 TO v
 150 INPUT "El numero ";(i);" (";(v$(i));"
)es ";v(i)
 160 NEXT i
 170 FOR i=1 TO v: FOR j=1 TO q: LET d(j)=
d(j)+v(i)*r(i,j): NEXT j: NEXT i
 210 FOR i=1 TO q
 220 IF d(i)>=d THEN  LET d=d(i): LET mi=i
 230 NEXT i
 240 CLS : INPUT "Es el resultado ";(mi);"
 (";(q$(mi));")? ";a$
 260 IF a$="s" THEN  LET s(mi)=1: LET sc=0
: FOR i=1 TO q: LET sc=sc+s(i): NEXT i: IF
 sc=q THEN  PRINT "Soy perfecto!": STOP
 265 IF a$="s" THEN  GO TO 128
 270 FOR i=1 TO q: PRINT i;" ";q$(i): NEXT
 i
 300 INPUT "Que resultado es? ";mj
 310 FOR i=1 TO q
 320 IF d(i)>=d AND i<>mj THEN : FOR j=1 T
O v: LET r(j,i)=r(j,i)-v(j): NEXT j
 330 NEXT i
 340 FOR j=1 TO v: LET r(j,mj)=r(j,mj)+v(j
): NEXT j
 370 PRINT "Antes de equivocarme habia con
seguido ";(sc);" aciertos!": PRINT : PRINT
 "Pulsa cualquier tecla para continuar"
 380 LET a$=INKEY$: IF a$="" THEN  GO TO 3
80
 390 FOR i=1 TO q: LET s(i)=0: NEXT i
 400 GO TO 128
```

4. CÓMO MEJORAR SU EXPERTO

4.1 DECISIONES PARALELAS Y SECUENCIALES

Ya hemos llegado a un punto en el que tenemos realmente un sistema experto. Después de un período de entrenamiento adecuado, el sistema puede tomar decisiones por nosotros sobre el tema que hayamos elegido, y, a menos que examinemos R(I,J), no necesitamos ni siquiera saber cómo lo hace.

Lo cual, en realidad, es suficiente para que la mayoría de las personas se sientan bastante satisfechas de sí mismas. Aparte, naturalmente, de los inevitables escépticos.

«Si fuese un verdadero sistema experto» —se quejarán estos escépticos— «no funcionaría así, sino de otro modo».

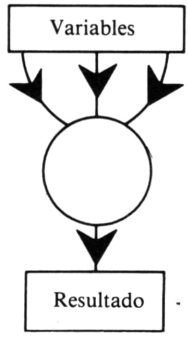
Y aducirán el hecho de que siempre que quieren una decisión experta han de teclear las respuestas a un montón de preguntas seguidas, antes de que el experto se digne hacer el menor trabajo.

Los sistemas expertos reales, argüirán, te hacen una pregunta, luego quizás otra, y, entonces, dependiendo del resultado, o bien te hacen más preguntas o bien te comunican su opinión experta. No empiezan por preguntarte absolutamente todo de una vez.

Lo que de hecho se está diciendo con todo ello, es que muchos sistemas utilizan procedimientos secuenciales de decisión, no procedimientos paralelos.

Y lo que nosotros tenemos es un procedimiento paralelo de decisión: recoge toda la información sobre un asunto, y entonces toma una decisión.

Los procedimientos paralelos consideran toda la evidencia antes de tomar una decisión



Los procedimientos secuenciales miran si pueden predecir un resultado después de cada nueva entrada. Si no pueden, piden más información

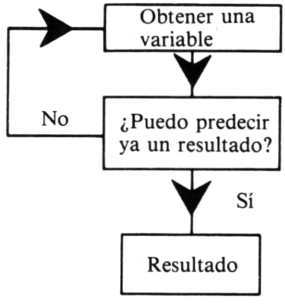


Fig. 4.1
Decisiones paralelas y secuenciales

El procedimiento secuencial se guía siempre por el último elemento de información que recibe. Así, en nuestro ejemplo del tiempo, un procedimiento secuencial podría decidir que lo más importante a saber para dar una opinión sobre el tiempo que va a hacer mañana es: ¿está lloviendo hoy? Haría entonces esa pregunta y luego pediría más información o bien emitiría un juicio.

Sin embargo, una de las dificultades con el ejemplo del tiempo es que no sabemos exactamente qué información es necesaria para predecir con exactitud si va a llover mañana. No ocurre así con el ejemplo del pájaro/avión: conocemos, por definición, las diferencias que hay entre

pájaros y aviones. Por tanto, veamos cómo resuelve este problema un experto secuencial.

El problema consiste en que usted piensa en un pájaro o en un avión y quiere que el experto averigüe de qué se trata.

«¿Tiene plumas?» —pregunta distraídamente el experto mientras da una chupada a su pipa.

«Efectivamente, sí» —reconoce usted.

E inmediatamente, el gran pensador deduce que es un pájaro. No ha tenido necesidad de considerar las seis variables conjuntamente.

Suponga ahora que piensa usted en un objeto, pero no ha especificado que puede volar. Los enfoques paralelo y secuencial para este problema son los indicados en la Figura 4.2.

«¿Tiene alas?» —le pregunta el experto—. Usted asiente, y enseguida el experto le plantea la pregunta anterior concerniente a las plumas, con cuya respuesta el experto toma la decisión. La técnica secuencial ha ganado.

En general, las técnicas secuenciales plantean menos preguntas que las paralelas, por lo que, en principio, pueden parecer más atractivas. O, en todo caso, menos charlatanas.

Pero así como el atractivo de los procedimientos secuenciales es evidente en el ejemplo del pájaro/avión, ya no lo es tanto en el ejemplo del tiempo.

Porque, aunque usted fuese un verdaero experto y conociese todo cuanto puede saberse sobre el tiempo, nunca podría estar absolutamente seguro de si va a llover mañana o no. Y, siendo así, el experto prudente acopiaría ante sí todos los indicios posibles antes de emitir ningún juicio. Y eso es lo que hacen los procedimientos paralelos: sabiendo que, en general, uno no puede estar absolutamente seguro del resultado, recopilan cuanta información pueden y hacen la mejor suposición posible. Un procedimiento secuencial pasa, en cuanto tiene la menor oportunidad, a las conclusiones, cosa que no hace nuestro sistema experto.

Si piensa usted un momento en ello, se dará cuenta de que la mejor actuación que puede tener un procedimiento secuencial sólo es tan buena como la de un procedimiento paralelo, porque como un procedimiento paralelo utiliza toda la información para llegar a una conclusión, siempre llega a la conclusión que tiene mayores posibilidades de

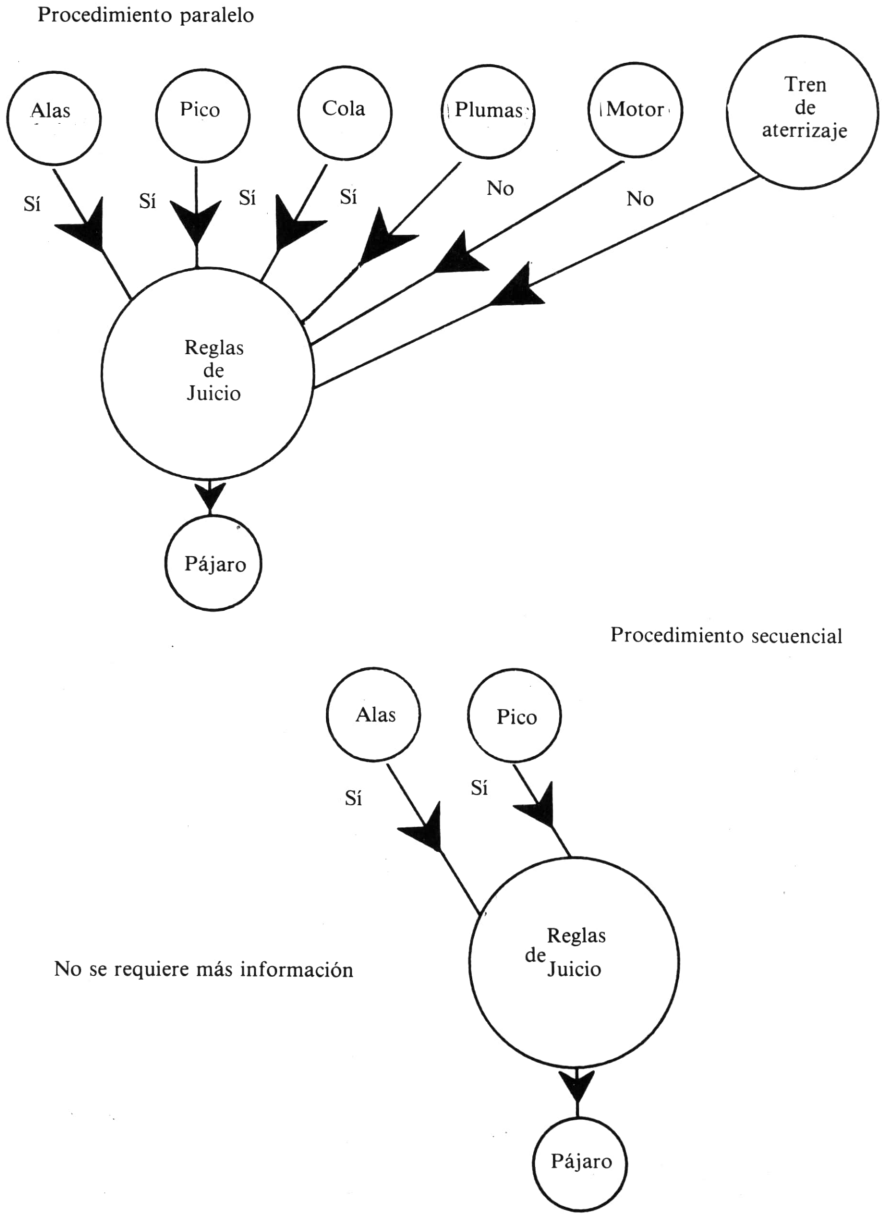


Fig. 4.2
Juicios paralelos y secuenciales

ser cierta. El procedimiento secuencial puede llegar a la misma conclusión, pasando secuencialmente a través de toda la información disponible, pero, al actuar así, hace exactamente lo mismo que el procedimiento paralelo, sólo que por etapas.

No siempre hay mucha diferencia entre los dos métodos. Supongamos que nuestro experto ha ido preguntando sobre las variables, borrando la pantalla después de cada respuesta. Entonces podría parecer que es un procedimiento secuencial, pero, como hasta el final no decide nada, en realidad es paralelo.

Evidentemente, podríamos reducir algo del esfuerzo de introducir información, si el experto, al disponer de un elemento de información decisivo, fuera capaz de saltarse el resto y dar una opinión. Pero, ¿cómo podemos escribir un programa así?.

Pues bien, si supiésemos sobre qué tema van a girar nuestras preguntas al experto, podríamos escribir un programa específico para ello. Pero, en nuestro caso, lo que queremos es un sistema experto de propósito general.

Y, si conociésemos las probabilidades exactas para cada resultado, también podríamos escribir un programa para hacerlo. Por ejemplo, conforme el experto recopilara, secuencialmente, elementos de información, podría explorar todas las posibilidades para encontrar la más probable, analizar todos los otros resultados para ver si podrían llegar a ser, dada la información actual, más probables que el que hasta este momento lo es, y, si ninguno pudiera llegar a ser más probable, el experto podría tomar una decisión sin necesidad de seguir; si alguno pudiera serlo, el experto necesitaría otro elemento de información.

Tomando el ejemplo del pájaro/avión, si hubiésemos introducido las probabilidades reales de los distintos resultados, en lugar de dejar que el experto las aprenda por sí mismo, entonces le habríamos dicho al experto que si el objeto tiene Plumas, es un Pájaro con probabilidad 1, y un Avión con probabilidad 0. Así, tan pronto como se le viesen (por así decirlo) las Plumas, el experto podría deducir que el único resultado posible es Pájaro, y dejar de hacer más preguntas tontas.

Pero no es así como hemos procedido. Por ejemplo, el experto aprende que sólo el resultado Pájaro es el que tiene Plumas. Pero también sabe algunas otras cosas. Por ejemplo, sabe que el resultado Pájaro tiene también la variable Pico. ¿Significa esto que sólo es un pájaro si tiene tanto plumas como pico, o que también lo es si sólo tiene una de esas cosas?. Y, más concretamente, ¿significa que en los ejemplos anteriores ocurrió por casualidad que los pájaros tenían pico o plumas, y

que podría haber un pájaro que no tuviese pico o plumas?. En otras palabras, ¿es algo absolutamente cierto, con probabilidad 1, o tiene una probabilidad muy grande, pero que no llega a 1, con lo cual podría aparecer un caso en el que la característica no se diera?

Pues bien, en lo que concierne a pájaros y aviones, podríamos comunicarle al experto esa información. Pero, ¿y en lo referente al tiempo?. En este caso, nada ocurre con probabilidad 1. Todo es bastante incierto, y nuestros conocimientos no son suficientes para decirle nada más al sistema experto. Nosotros mismos no conocemos las respuestas, salvo si esperamos para ver cómo evoluciona el tiempo. Por tanto, ¿cómo explicárselo al sistema experto?

En casos específicos puede idearse un procedimiento secuencial para ahorrar tiempo y molestias y que puede dar juicios tan buenos como los de un procedimiento paralelo. Pero, en general, un procedimiento paralelo tomará decisiones tan buenas, y normalmente mejores que uno secuencial, y el programa que utiliza procedimientos paralelos es de propósito mucho más general, porque hace menos suposiciones.

Dicho esto, consideremos cómo podríamos convertir nuestro sistema experto en un mecanismo secuencial, como muestra la Figura 4.3.

Supongamos que el sistema presenta el nombre de la primera variable, y pide un valor para ella, que hemos de darle. Entonces el sistema puede calcular los valores iniciales de D para los Q resultados, utilizando este valor de una sola variable.

Hecho esto, el sistema puede explorar esos valores de D, seleccionar el mayor y considerar si podría ser o no el resultado correcto. Para esto último, lo que puede hacer es ir explorando todas las reglas en R(I,J) para ver si alguna de ellas podría incrementar su propio valor de D lo suficiente para superar al del que actualmente se considera como más probable. Pero el hacer esto implica aventurar valores ficticios de V(I) para trabajar con ellos. Ahora bien, si el sistema verdaderamente los aventurase, estaría introduciéndose más incertidumbre en la situación de la que ya hay. Si tiene que hacerlo, y estar seguro de lo que hace, el sistema tiene que conocer los valores máximo y mínimo que puede tomar V(I) para cada una de las variables. Si podemos darle esos valores, el sistema experto puede, al menos, descartar algunas de las opciones, y, si puede descartar todas menos una, entonces puede tomar una decisión.

Pero, ¿qué pasa si toma una decisión incorrecta?.

Pues bien, si se le ha dado la información correcta sobre los valores máximos y mínimos de las variables, sólo puede tomar una decisión in-

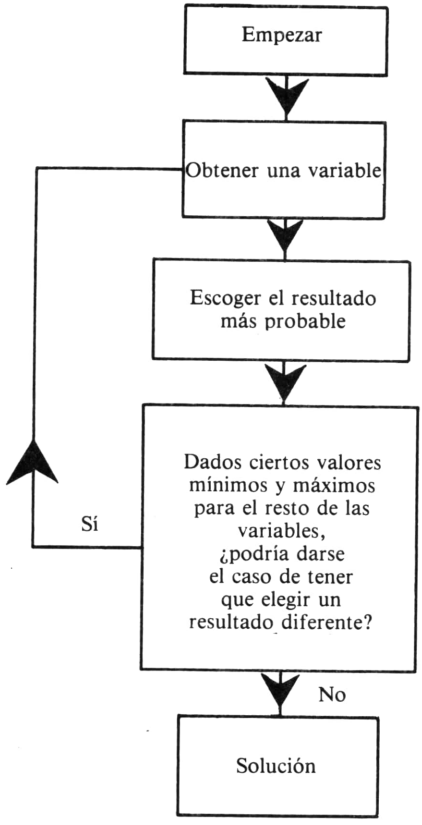


Fig. 4.3
Un algoritmo experto secuencial

correcta si sus reglas contenidas en R(I,J) son falsas. Y podría ser que las reglas tuvieran que modificarse en todas las variables. Por tanto, el programa tendría que volver al modo paralelo para que le diésemos todos los valores de las variables y ajustar R(I,J), tal como lo hacía en el caso de procesamiento paralelo.

En lo que se refiere a la construcción de su propio sistema experto, puede que no valga realmente la pena el esfuerzo para hacer todo esto. Especialmente, si a usted no le gusta la programación.

Pero si queremos hacer que el sistema sea más interesante o más fácil de usar, puede que sí valga la pena. Si ello va a conducir o no a al-

guna diferencia en el funcionamiento del experto, va a depender de los valores máximos y mínimos que proporcione usted. Las diferencias entre los maximos y mínimos deben ser lo suficientemente grandes para que constituyan verdaderos límites de los datos de entrada; si no, el experto va a cometer errores. Pero si las diferencias fueran excesivamente grandes, no ayudarían apenas al experto a concretar una decisión, y funcionarían de nuevo lo mismo que un procesador paralelo (¡pero tras un esfuerzo mucho mayor de programación por su parte!).

4.2 CÓMO INCLUIR ALGO DE SENTIDO COMÚN

Como hemos visto, si no podemos decirle al experto nada más sobre el problema a tratar, los procedimientos paralelos que hemos descrito dan la mejor solución posible hasta el momento. Pero si por lo menos podemos especificar valores máximos y mínimos para las variables, el sistema podrá, en algunas circunstancias, omitir algunos elementos cuando solicite información.

Vamos a bosquejar el método en BASIC Applesoft paso a paso. Supongamos que DIMensionamos M(2,V) para que contenga los valores máximo y mínimo de cada una de las variables, los mínimos en M(1,I) y los máximos en M(2,I). Podemos disponer la introducción de esos valores al comienzo de la sesión de entrenamiento del programa, ya sea inmediatamente después de la introducción de cada nombre de variable, o como un bloque separado de información, posteriormente. También hay que DIMensionar VC(V) y PD(Q).

Entonces, para entrar en una sesión de entrenamiento, tendremos:

```
 2  V=6:Q=3
 5  REM: SE PRECISAN VALORES PARA V Y Q
10  DIM VC(V),PD(Q),M(2,V),D(1),V(V),R(V,Q): REM: HAY QUE
    DAR VALORES MINIMOS Y MAXIMOS EN M(2,V)
20  FOR I=1 TO V
30  VC(I)=1
40  NEXT
50  REM: CON ELLO SE INICIALIZA VC, UN VECTOR QUE
    SEÑALA LAS VARIABLES QUE TODAVIA NO HAN SIDO
    UTILIZADAS EN EL PROCESO DE DECISION
60  FOR J=1 TO Q
70  D(J)=0
80  NEXT
90  REM: CON LO ANTERIOR SE PONE A CERO EL VECTOR D
```

```
100 FOR VV=1 TO V
110 INPUT V(VV): REM: EL EXPERTO OBTIENE UN VALOR
    DE UNA VARIABLE
120 VC(VV)=0: REM: SE PONE A CERO LA SEÑAL DE LA
    VARIABLE
130 REM: AHORA SE CALCULA, EN LO POSIBLE, EL
    VECTOR D
140 FOR J=1 TO Q
150 D(J )=D(J)+V(VV)*R(VV,J)
160 NEXT
170 REM: CON ELLO SE ACTUALIZA EL VECTOR DE
    DECISION
180 D=0
190 FOR J=1 TO Q
200 IF D(J) >=D THEN: D=D(J): MJ=J
210 NEXT: REM: SE BUSCA EL MAXIMO EN D(J) Y SE PONE
    EN D. MJ ES LA DECISION MAS ADECUADA HASTA EL
    MOMENTO
220 FOR J=1 TO Q
230 PD(J)=D(J)
240 NEXT
250 REM: EL VECTOR PD GUARDA LOS POSIBLES VALORES
    DE DECISION, UTILIZANDO LAS VARIABLES QUE YA SE
    HAN INTRODUCIDO
260 FOR J=1 TO Q
270 FOR I=1 TO V
280 IF VC(I)=1 THEN: IF R(I,J) >=R(I,MJ) THEN:
    PD(J)=PD(J)+(R(I,J)—R(I,MJ)*M(2,I)
290 IF VC(I)=1 THEN: IF R(I,J) < R(I,MJ) THEN:
    PD(J)=PD(J)+(R(I,J)—R(I,MJ))*M(1,I)
300 NEXT: NEXT
```

(Lo que hacen las cinco últimas líneas es recorrer todas las variables que aún no hemos usado. Utilizando los valores máximos y mínimos conocidos, se altera PD(J), intentando encontrar otro posible resultado en lugar de PD(MJ). El programa hace esto suponiendo que todas las variables que faltan actuarán en contra de la elección actual. Como sólo se trata de ver si esta elección actual podría resultar reemplazada, el programa trabaja con la diferencia R(I,J)-R(I,MJ) y no con R(I,J) y R(I,MJ) separadamente).

```
310 M2=PD(MJ)
320 FOR J=1 TO Q
330 IF PD(J)>=M2 THEN: M2=PD(J): MY=J
340 NEXT.
```

```
350 IF MY=MJ THEN: STOP: REM: NADA CONTRADICE EL
    RESULTADO
360 IF MY<>MJ: NEXT VV: REM: AUN NO ESTA CLARO, Y SE
    NECESITA OTRA VARIABLE
```

Si después de todo esto se toma una decisión incorrecta, el programa deberá comprobar VC(I), para ver qué valores de V(I) no se han introducido todavía, y se dan esos valores. Luego, bifurcará a la rutina para modificar R(I,J).

El método de buscar ordenadamente los valores de V(I), puede que no sea el más lógico. Por ejemplo, si todos los valores de R(I,J) fueran iguales para una determinada variable, entonces el obtener el valor de esa variable no le diría nada al sistema experto; podría saltársela tranquilamente. Para considerar este caso, podríamos comprobar, con unas pocas líneas de programa, antes de solicitar una variable, si todos los R(I,J) son iguales para esa variable, y, sí lo son, el experto podría saltar a la siguiente variable sin pedir ningún valor.

Lo que, en realidad, necesitamos, es un método general para decidir cuál es la variable cuyo valor debe pedir el sistema en cada momento. Por ejemplo, algunas variables podrían afectar más que otras al resultado. Lo que sugiere el párrafo anterior es un método que separa las variables en dos grupos: las que pueden afectar al resultado y las que no pueden hacerlo. Pero si escribimos una rutina para tratar de conseguir primero aquellas variables que parezcan afectar más y con mayor probabilidad al resultado, seguiremos teniendo la posibilidad de que, al final, el sistema pregunte por todas las variables que no afectan de ningún modo al resultado, y, así, todo lo que habremos ganado es una reordenación de las variables, y no una reducción en el número de ellas a considerar.

Un método que puede probarse consiste en considerar primero aquellos valores de variables que tienen mayor margen de variación, en función de sus valores máximos y mínimos (supuesto que podemos proporcionar tales valores). Como esas variables son las que más pueden variar, podría presumirse que también son las que más contribuyen al proceso de toma de decisión. Evidentemente, en el ejemplo del pájaro/avión, los valores máximo y mínimo son 1 y 0 respectivamente, correspondientes a la presencia o ausencia de cada característica, por lo que esto no nos ayuda mucho. Sin embargo, cuando trabajamos con datos relativos, por ejemplo, a las condiciones meteorológicas, podemos tener distintas variaciones en los valores máximos y mínimos para los diferentes elementos. La nubosidad podría ir de cero (nubosidad nula) a 10 (10 décimas partes de cielo cubierto), y la precipitación podría variar entre 0 y 5, medida en decalitros por metro cuadrado. Podría entonces

parecer que la nubosidad, que tiene mayor margen, debería ser la primera entre esas dos variables a comprobar. A menos, naturalmente, que decidamos medir la precipitación en litros, en cuyo caso el margen de valores es de 0 a 50 litros, y la cantidad de lluvia sería la primera variable a considerar.

Todo esto sirve para ilustrar algo más el hecho de que, en general, el método de trabajo del sistema experto no se puede concretar demasiado, porque si tiene que ser totalmente general con relación a los datos que le damos, no podemos hacerlo demasiado específico en cuanto a la forma en que maneja esos datos.

Si construye usted su sistema experto para que sea totalmente independiente del campo concreto en el que se supone que ha de adquirir experiencia, entonces, en ciertos casos, es probable que parezca algo torpe y lento. Si tal cosa ocurre, puede que le apetezca construir el sistema a la medida de una aplicación específica, haciendo, por ejemplo, que pregunte al principio del proceso por ciertas variables y que omita otras, de acuerdo con el conocimiento previo que usted tiene sobre la aplicación. Pero si lo hace así, descrubrirá probablemente que su sistema experto no puede aprender tan rápidamente otros problemas. Ya no será un sistema experto de propósito general, aunque puede que sea mejor para el propósito que ha sido construido.

En realidad, todo depende de para qué quiere usted su sistema experto. Si sabe para qué lo quiere, entonces puede construir un sistema que para ese propósito es mejor que cualquier sistema de propósito general que le den.

Dicho esto, podemos, en cualquier caso, intentar hacer que el sistema busque primero las «mejores» variables. Podemos hacerlo considerando cuánta variación pueden originar las variables en la regla de decisión.

Si DIMensionamos VR(V) para guardar un indicador de esta variación (los «Valores de Regla»), podemos escribir:

```
10  FOR I=1 TO V
20  FOR J=1 TO Q
30  VR(I)=VR(I)+ABS(M(1,I)-M(2,I))*ABS(R(I,J))
40  NEXT: NEXT
```

Con ello, calculamos el valor absoluto de la diferencia entre los valores máximo y mínimo de cada variable, y multiplicamos esta diferencia por los valores absolutos de las reglas para esa variable. Entonces, podríamos explorar VR(I) para buscar qué variable corresponde al ma-

yor valor de VR(I). Esta sería la variable que suponemos más importante, por lo que su valor se introduciría primero. Si el sistema hubiera de preguntar por otra variable, lo haría por aquella que tuviese el siguiente valor más alto en VR(I), y así sucesivamente.

El que este método funcione bien depende, aparte de los comentarios ya hechos, de la precisión con la que conocemos los valores máximos y mínimos de las variables y del grado de bondad de las reglas desarrolladas por el sistema hasta ese momento.

Evidentemente, cuando el sistema empieza y todos los valores de R(I,J) son nulos, no tiene ni idea de qué variables son importantes para el problema y cuáles no. Si la experiencia va a mejorar o no el sistema en este aspecto, depende principalmente del problema que se le ha dado a resolver.

Hay otro refinamiento que puede utilizarse cuando se intenta decidir por cuál variable preguntar. Nuestro método consiste en escoger la variable que tenga el máximo valor en VR(I). Supongamos que ya se ha introducido el valor de esa variable. Entonces, se hace VC(I) = 0 para indicar que ya no hay que preguntar por ella.

El sistema, entonces, pasa a comprobar si puede deducir un resultado seguro. Si no puede, preguntará por el valor de otra variable, aquélla que tenga el siguiente valor más grande en VR(I).

Pero cuando el sistema comprueba si hay otros resultados posibles, puede ocurrir que sólo queden unos pocos resultados como posibles candidatos. Comprobando los valores posibles de las reglas de decisión en PD(J), algunos de ellos puede que sean menores que el D(J) correspondiente al resultado que el sistema ha elegido hasta ahora como más probable. En este caso, esos resultados ya no son candidatos posibles en ninguna circunstancia, y pueden ignorarse a todos los efectos.

Supongamos que DIMensionamos QC(Q) para los resultados, y utilizamos este vector exactamente del mismo modo que hemos utilizado VC(V) para las variables. Inicializamos con 1 todos sus elementos y, en el momento en que se vea que el resultado J no puede ser cierto, hacemos QC(J) = 0.

Entonces, siempre que nos enfrentamos a un conjunto de valores, rápidamente podemos eliminar los que están «muertos». Por ejemplo:

```
10  FOR I=1 TO V
20  FOR J=1 TO Q
30   VR(I)=VR(I)+ABS(M(1,I)-M(2,I))*ABS(R(I,J))*VC(I)
```

```
    *QC(J)
40  NEXT:NEXT
```

Con ello, no se cambia nada en el caso de que todas las variables y todos los resultados estén aún activos, porque VC(I) y QC(J) serán todos iguales a 1. Pero si cualquier variable ya se ha utilizado o cualquier resultado ya no es posible, entonces VC(I) o QC(J) son iguales a 0, y VR(I) queda inalterado en lo que respecta a esa variable o resultado. Ello significa simplemente que estamos considerando el problema reducido de cómo seguir en el caso de que algunos de los elementos ya no estén activos.

Por otra parte, el uso de los valores de VR(I) que hemos seguido aquí no es, en absoluto, obligatorio. Usted puede, por ejemplo, elevar al cuadrado QR(I,J), en lugar de usar su valor absoluto. Todo depende de lo que piense usted que puede ser más útil. Por ejemplo, supongamos que quisiera usted elegir entre dos variables, con dos resultados y los siguientes valores de R(I,J):

	Resultado 1	Resultado 2
Variable 1	2	2
Variable 2	1	3

Ahora bien, si ambas variables tuviesen los mismos valores máximo y mínimo, ambas darían los mismos valores en VR(I), porque cada una de ellas suma 4. Pero usted preferiría que la variable 2 se investigase primero, porque es la que tiene un efecto mayor (realmente, en este ejemplo, es la única que tiene algún efecto). Si usted hubiese tomado R(I,J) elevado al cuadrado, habría obtenido:

$$VR(1) = 2^2 + 2^2 = 8$$
$$VR(2) = 1^2 + 3^2 = 10$$

con lo que habría elegido primero la variable 2.

Naturalmente, la variable 1 habría quedado en cualquier caso fuera de juego, porque tiene el mismo valor para cada resultado. Pero suponga que existe otra variable.

	Resultado 1	Resultado 2
Variable 3	0	4

Esto daría un valor de 16 para VR(3), por lo que se seleccionaría antes que la variable 2 (o la 1). Si se hubiese tomado el valor absoluto de R(I,J) en VR(I), no habría resultado ninguna diferencia entre las tres va-

riables, aunque, como puede usted ver, sus contribuciones al problema son diferentes.

Del mismo modo que el método anterior no era en absoluto obligatorio, tampoco lo es éste. Elevar al cuadrado R(I,J) acentúa cualquier diferencia que pueda existir. Pero tomar una potencia mayor la acentuaría aún más. El tomar una potencia impar (y no el valor absoluto) mantendría intacto el signo positivo o negativo de la diferencia, lo cual podría ser útil (francamente, no sabemos para qué), y el tomar cualquier otra función, podría hacer que todo funcionase de una manera un tanto extraña, que podría ser conveniente para algún problema raro.

Desde luego, hay un modo de hacer las cosas que es todavía más complicado, pero que en ciertos casos podría valer la pena adoptar. Consiste en lo siguiente: Considerar la suma de los cuadrados de las desviaciones de cada una de las reglas de variables en torno a su media. Para ello, se calcula el valor medio de las reglas para cada variable, y luego se calcula el cuadrado de la diferencia entre las reglas y esos valores. Por ejemplo:

```
10  FOR J=1 TO Q
20  M=M+(R(I,J)/Q)*VC(I)*QC(J)
30  NEXT
40  FOR J=1 TO Q
50  VR(I)=VR(I)+((R(I,J)-M)^2)*ABS(M(1,I)-M(2,I))*VC(I)
    *QC(I)
60  NEXT
```

Con ello se obtiene VR(I), para la variable I, como suma de los cuadrados de las desviaciones en torno al valor medio de R(I,J), considerando todos los resultados posibles.

Habría que realizar el cálculo para cada I, incluyendo siempre VC(I) y QC(J) cuando se calculan las medias y la suma de los cuadrados, si se quieren tener en cuenta las variables que ya se han dado y los resultados que ya no son posibles.

Lo importante a señalar sobre este método es que podríamos tener las siguientes reglas:

	Resultado 1	Resultado 2
Variable 1	1	2
Variable 2	1	—2

con las que, intuitivamente, se observa que la variable 2 es la más impor-

tante, debido a la gran diferencia que puede originar en el resultado. Pero ABS(R(I,J)) no detectaría ninguna diferencia entre los dos conjuntos, y tampoco lo haría R(I,J) elevado al cuadrado. Las reglas de la variable 2 son las que tienen una mayor variación alrededor de su valor medio, y esta variación es la que podemos tratar de medir para seleccionar la mejor variable a considerar.

La variable 1 tiene un valor medio de 1.5 y la 2, —0.5, por lo que podemos calcular, respectivamente:

$$(1-1.5)^2+(2-1.5)^2=0.5, \text{ y } (1-(-0.5))^2+(-2-(-0.5))^2=4.5$$

Con este método se elige la variable 2. Y si merece o no la pena depende sobre todo de cuántas variables haya para considerar. Si son muchas, cualquier método que ayude a elegir primero la más adecuada puede ser bueno.

4.3 UNA PRUEBA PARA NUESTRO NUEVO EXPERTO

Vamos a modificar nuestro programa original para que trate de eliminar algunas variables utilizando el método del máximo/mínimo explicado en el Apartado 4.2. El listado modificado se encuentra al final de este Apartado.

Para ver si el experto funciona bien o mal, le daremos el ejemplo del pájaro/avión/planeador. Y para que el análisis no resulte excesivamente complicado, sólo le daremos tres elementos de información: alas, pico y motor.

Un pájaro tiene alas y pico, pero no motor. Un avión tiene alas y motor, pero no pico. Un planeador tiene alas, pero no pico ni motor.

Al empezar, la matriz de reglas, R(I,J), contiene ceros. Le decimos al sistema experto que el problema tiene tres variables y tres resultados, y le damos los nombres de esas variables y de esos resultados. Luego damos valores máximos y mínimos para cada una de las tres variables. Naturalmente, en este caso, son ceros y unos, correspondiendo a la presencia o ausencia de cada característica:

Variable	Valor mínimo	Valor máximo
Alas	0	1
Pico	0	1
Motor	0	1

En el momento en que introduzcamos una descripción completa de cada uno de los tres resultados, la descripción se guardará en V(I), y será:

Variable	Pájaro	Avión	Planeador
Alas	1	1	1
Pico	1	0	0
Motor	0	1	0

La sucesión de acontecimientos depende ahora del usuario (usted), pero supongamos que todo ocurre del siguiente modo:

Pensamos en un pájaro, e introducimos los valores de V(I) correspondientes a un pájaro. La matriz de reglas R(I,J) se ajusta, porque el experto da un resultado falso:

R(I,J)	Pájaro	Avión	Planeador
Alas	1	—1	—1
Pico	1	—1	—1
Motor	0	0	0

Es decir, el V(I) de pájaro se ha sumado a la regla del resultado pájaro y se ha restado de las otras reglas.

El experto pasa a otra sesión. De nuevo pensamos en un pájaro. Pregunta por el valor de «Alas», y respondemos con «1». El sistema pregunta entonces si puede deducir que el resultado es pájaro. Respondemos que sí, y el sistema no tiene que ajustar nada.

Ahora bien, obviamente, sabemos que las alas no son suficientes para identificar a un pájaro. En realidad, no representan en absoluto una evidencia de que lo sea. Pero en el estado actual de conocimientos del experto, es suficiente. Porque con V(I) = (1,x,x), las reglas dan un valor mayor para pájaro que para cualquier otro resultado. Y, teniendo en cuenta la información que tiene el sistema sobre máximos y mínimos, no hay valores para las otras dos variables que puedan alterar esta decisión.

Por tanto, el experto ha obtenido la respuesta correcta con información imperfecta; no porque funcione incorrectamente, sino porque sus reglas son, de momento, imperfectas. Podemos ser un poco comprensivos con él: después de todo, es la primera vez en su vida que se tropieza con un pájaro.

Vayamos a otra sesión. Ahora pensamos en un avión. El sistema pide un valor para «alas», y respondemos con «1». Pregunta de nuevo si

puede deducir el resultado pájaro. Se trata de una pregunta razonable, porque la vez anterior podía. Le contestamos que no puede.

El sistema pregunta qué resultado debería haber sido, y le respondemos que Avión. Entonces, pide valores para el resto de V(I), para poder reajustar sus reglas. Le damos los valores correctos para pico y motor: V(I) = (1,0,1). Y el sistema ajusta sus reglas.

Otra sesión. De nuevo pensamos en un avión. El sistema pide primero un valor de «Pico». Le damos «0». Luego pregunta sobre «Motor». Damos «1». Entonces pregunta si puede deducir el resultado avión. Y sí puede.

Esto es bastante halagüeño. Esta vez no ha preguntado sobre las alas lo cual está bien, porque la respuesta no le habría dicho nada.

Observemos la matriz de reglas, R(I,J), tal como está ahora:

	Pájaro	Avión	Planeador
Alas	0	0	—2
Pico	1	—1	—1
Motor	—1	1	—0

Después del último error cometido, el experto ha sumado las variables de avión a la regla de avión y los ha restado de las reglas de pájaro y planeador.

Ahora bien, ¿por qué ha preguntado sobre el pico en primer lugar?

Pues lo ha hecho porque ésta parecía ser la variable más importante. En cada regla, hay un 1 o un —1 asignado a pico, de modo que sumando los productos de los valores máximos de variables por los valores absolutos en la matriz de reglas, el experto valora pico con 3, motor con 3 y alas con 2 en orden de importancia. Como pico está antes que motor en la lista de variables, el sistema pregunta primero por el pico.

A partir de ahora, si sólo se consideran pájaros y aviones, el experto acertará siempre. En cada ocasión, preguntará primero sobre el pico. Si lo tiene, dirá que es un pájaro. Si no lo tiene , preguntará sobre el motor, y, si tiene motor, decidirá que es un avión.

Entremos en otra sesión, en la que pensamos en un planeador. Al preguntarnos sobre el pico, respondemos con «0». El sistema pregunta entonces sobre el «motor», respondemos con «0», y sobre «alas», y respondemos con «1».

En esta situación, el sistema aventura que se trata de otro avión, lo que no es cierto, y así se lo decimos: le decimos que se trata de un planeador, y el sistema vuelve a ajustar la matriz R.

Otra sesión. Ahora pensamos en un pájaro. El experto pregunta sobre las alas y pico. Respondemos «1» y «1», y decide, correctamente, que es un pájaro.

Otra sesión. Pensamos en un planeador. Pregunta sobre las alas, pico y motor, y respondemos con «1», «0» y «0». El experto aventura planeador, lo que es correcto.

Otra sesión, en la que pensamos en un avión. Pregunta sobre alas, pico y motor, y respondemos «1», «0» y «1». El sistema deduce, correctamente, que es un avión.

Otra sesión, y ahora pensamos en un pájaro. Pregunta sólo sobre alas y pico, y respondemos «1» y «1», y el sistema acierta de nuevo.

Después de esto, el sistema es ya perfecto y no comete más errores. Puede distinguir correctamente entre pájaros, aviones y planeadores, y para ello no necesitaba preguntar siempre por las tres variables.

El conjunto de reglas en este momento es:

	Pájaro	Avión	Planeador
Alas	—1	—1	—1
Pico	1	—1	—1
Motor	—1	1	—1

Es conveniente hacer algunas observaciones sobre estas reglas que en este momento tiene el sistema.

En primer lugar, el sistema identifica al planeador sólo por defecto, por así decirlo. Es decir, el planeador tiene como valores (1,0,0), con los que cualquiera de los tres resultados da un valor de decisión de —1; no hay razón para elegir planeador en lugar de cualquier otro resultado. La razón por la que el sistema elige planeador es que, habiendo visto que el objeto tiene alas y no tiene pico, busca el valor de decisión máximo; como todos son iguales, se queda con planeador, que corresponde al último valor examinado. Cuando el sistema encuentra que el objeto tampoco tiene motor, hace lo mismo: se queda con planeador como el último elemento de la lista, y encuentra que nada puede reemplazar a este juicio, porque no hay un valor mayor posible.

En un ejemplo como éste, el resultado depende mucho de que el

ajuste de las reglas y la toma de decisiones se hagan sobre la base de comprobaciones condicionales del tipo «mayor que» o del tipo «mayor o igual que». La comprobación más exigente (mayor o igual que) altera las reglas más a menudo y conduce inicialmente a más errores. Pero con frecuencia lleva a un conjunto de reglas algo mejor, en cuanto a que se plantean menos preguntas siempre que es posible. Una comprobación del tipo «mayor que» modifica menos frecuentemente las reglas, y funciona, en parte, guiado por el orden natural de los datos en la lista (lo cual es algo que debería usted recordar cuando intente averiguar qué es lo que ha hecho el programa con sus reglas).

Otra observación es que el experto suele preguntar sobre Alas, y sabemos que esta cuestión no tiene en absoluto nada que ver con el resultado. La razón por la que lo hace es porque los valores de regla en el vector VR son (3,3,3) para (alas, pico, motor). Y esto nos hace pensar que podría haber sido preferible algún otro método para calcular los valores de regla. Por ejemplo, si calculásemos las medias de las variables, (—1,—1/3,—1/3), podríamos calcular la suma de los cuadrados de las diferencias entre los valores de cada variable y su media (0, 2.67, 2.67), lo que claramente da pico y motor como variables más importantes, y manda las alas, por decirlo de algún modo, a tomar viento. (A propósito, la suma de los cuadrados de las diferencias de los valores de una variable con respecto a su valor medio es lo mismo que la varianza de la variable, sin dividir por n; si no lo ve muy claro, acuda al Sumario Técnico, al final del libro).

Por otra parte, si hubiésemos entrenado al experto presentándole los problemas en otro orden, habríamos obtenido muy probablemente otro conjunto diferente de reglas.

Pero todo lo que pedimos de un conjunto de reglas para que sea aceptable es que funcionen, y las que se han obtenido lo hacen.

Recuerde nuestra charla sobre separabilidad lineal. Hemos conseguido un sistema experto que se las ha ingeniado para trazar líneas entre los tres grupos de objetos: pájaros, aviones y planeadores. No importa dónde se encuentran exactamente las líneas, siempre y cuando sirvan para separar los tres grupos. El conjunto de reglas actual lo hace así, pero no sería difícil pensar en otro que también lo hiciese.

Este ejemplo está muy bien, pero no hay que suponer que todo problema que se le plantee al experto va a ser así de fácil.

Supongamos que mantenemos los tres resultados, pájaro, avión y planeador, pero damos seis variables: alas, cola, pico, plumas, motor, y tren de aterrizaje. El sentido común nos dice que la mayor parte de esas

variables son redundantes: no son necesarias para poder responder a la pregunta básica. Pero ¿va a averiguarlo el experto?.

La respuesta es: No.

Puesto que sólo hay tres resultados, sabemos que dos preguntas bien elegidas bastarían para resolver el asunto. Sin embargo, el experto, después de haber sido suficientemente entrenado, pregunta sobre los siguientes elementos para cada uno de los tres resultados:

Pájaro	Motor-Respuesta: 0 Tren de aterrizaje-Respuesta:0 Pico-Respuesta: 1	Deducción: Pájaro
Avión	Motor-Respuesta: 1 Tren de aterrizaje-Respuesta: 1	Deducción: Avión
Planeador	Motor-Respuesta: 0 Tren de aterrizaje-Respuesta: 0 Pico-Respuesta: 0 Plumas-Respuesta: 0	Deducción: Planeador

Ahora bien, esto no está tan mal. El sistema acierta siempre, y no pregunta sobre todas las variables.Por ejemplo, nunca pregunta sobre alas ni cola, lo cual está bien, porque esas variables no le dicen nada en absoluto. Sin embargo, desde nuestro punto de vista, es algo duro de mollera, pues pregunta siempre sobre el tren de aterrizaje inmediatamente después de preguntar sobre motor. Pero esto es debido a que nosotros sabemos que, en estos ejemplos, los dos siempre van juntos. Asimismo, suele preguntar sobre las plumas después de hacerlo sobre el pico, otros dos elementos que nosotros sabemos que siempre van a darse juntos.

Lo importante es, desde luego, que el sistema experto no sabe que esas cosas siempre se dan conjuntamente. Una vez que ha preguntado sobre el motor, una cuestión que a él le parece importante, encuentra que no puede estar seguro del resultado, y tiene que hacer otra pregunta. El tren de aterrizaje le parece importante, por lo que pregunta por él. No se da cuenta que, a partir de la respuesta a motor, podría haber deducido la de tren de aterrizaje. Por lo que el experto sabe, motor y tren de aterrizaje podrían haber aparecido siempre juntos hasta ahora por mera casualidad. Podría darse un caso en que no fuera sí, y ello habría cambiado su opinión sobre el resultado posible. El sistema necesita seguir preguntando hasta que, dado el estado de sus reglas y los valores máximos y mínimos asignados a las variables, no haya posibilidad algu-

na de conseguir nueva información sobre alguna variable que le pueda hacer cambiar su parecer sobre el resultado más verosímil.

El saber que el objeto tenía un motor no era suficiente para que el experto estuviese absolutamente seguro del resultado. Suponga (imagínese que el sistema está pensando por sí mismo) que yo tuviese un objeto con motor y sin tren de aterrizaje. ¿Me haría ello pensar de otro modo sobre la cosa de que se trata?. Sí, desde luego. Mejor preguntar sobre el tren de aterrizaje.

Qué tipo de objeto piensa exactamente el experto que puede tener motor y no tener tren de aterrizaje, es un misterio. Quizás podríamos haber incluido el resultado «Cohete». Aunque sólo fuera para hacer feliz al experto.

A continuación, damos un listado del programa utilizado en este apartado. Hace uso del algoritmo de aprendizaje. Se utilizan valores máximos y mínimos para tratar de llegar rápidamente a una conclusión. Las preguntas se plantean sobre la base de sus valores, utilizando la función ABS con los valores máximos y mínimos para tratar de valorar su importancia. Los resultados que no pueden alcanzar un valor suficientemente alto para desplazar al que actualmente es el mejor, se eliminan de los cálculos posteriores.

Después de que usted haya ejecutado este programa, pruebe a modificarlo de modo que los valores de regla en VR se calculen utilizando la suma de cuadrados de las diferencias con las medias (ver pág. 96) en lugar de la función ABS: ello influirá en el orden en que el sistema plantea las preguntas.

Fig 4.4
Programa modificado con máximos y mínimos

Listado para el Apple II

```
20  HOME : INPUT "CUANTAS VARIABLES
    HAY? ";V
30  DIM V(V),V$(V),M(2,V),VC(V),PD(V
    ),VR(V)
40  PRINT "INTRODUZCA SUS NOMBRES"
50  FOR I = 1 TO V
60  PRINT "LA VARIABLE ";I;" ES ";: INPUT
    V$(I)
70  INPUT "SU VALOR MINIMO ES= ";M(1
```

```
    ,I): INPUT "Y SU VALOR MAXIMO= "
    ;M(2,I)
80  NEXT I
90  INPUT "CUANTOS RESULTADOS HAY? "
    ;Q: DIM Q$(Q),R(V,Q),D(Q),QC(Q)
100  PRINT "INTRODUZCA SUS NOMBRES"
110  FOR I = 1 TO Q
120  PRINT "EL RESULTADO ";I;" ES ";
    : INPUT Q$(I)
130  NEXT I
140  PRINT "VARIABLE","VALOR MIN"; SPC(
    5);"VALOR MAX": FOR I = 1 TO V: PRINT
    V$(I),M(1,I),M(2,I): NEXT I: PRINT
    "PULSE CUALQUIER TECLA PARA SEGU
    IR": GET X$
150  HOME : PRINT "ENTRO EN UNA SESI
    ON DE ENTRENAMIENTO": PRINT "DEM
    E VALORES DE LAS VARIABLES": PRINT
    "Y YO LE DARE UN RESULTADO": PRINT
    "USTED ME DIRA SI ES CORRECTO O
    NO"
160  FOR I = 1 TO V:VC(I) = 1:V(I) =
    0: NEXT I: FOR J = 1 TO Q:D(J) =
    0:PD(J) = 0:QC(J) = 1: NEXT J
170 VR = 0:VV = 1: FOR I = 1 TO V:VR
    (I) = 0: FOR J = 1 TO Q:VR(I) =
    VR(I) +  ABS (M(1,I) - M(2,I)) *
     ABS (R(I,J)) * VC(I) * QC(J): NEXT
    J: IF VR(I) > VR THEN VV = I:VR =
    VR(I)
180  NEXT I: REM  SE ESCOGE LA VARIA
    BLE MAS INFLUYENTE
190  IF VR = 0 THEN  GOTO 390: REM
    LA VARIABLE MAS INFLUYENTE NO CO
    NTRIBUYE EN NADA
200  PRINT "LA VARIABLE ";VV;" (";V$
    (VV);") VALE ";: INPUT V(VV)
210 VC(VV) = 0: REM  SE PONE A CERO
    LA SEÑAL
220  FOR J = 1 TO Q
230 D(J) = D(J) + V(VV) * R(VV,J)
```

```
240  NEXT J: REM  SE ACTUALIZA EL VE
     CTOR DE DECISION
250 D = D(1): FOR J = 1 TO Q
260  IF D(J) >  = D THEN D = D(J):MJ
      = J
270 PD(J) = D(J): NEXT J: REM  SE OB
     TIENE EL RESULTADO MAS PLAUSIBLE
      DE MOMENTO
280  FOR J = 1 TO Q
290  FOR I = 1 TO V
300  IF VC(I) = 1 THEN  IF R(I,J) >
     R(I,MJ) THEN PD(J) = PD(J) + (R(
     I,J) - R(I,MJ)) * M(2,I)
310  IF VC(I) = 1 THEN  IF R(I,J) <
     R(I,MJ) THEN PD(J) = PD(J) + (R(
     I,J) - R(I,MJ)) * M(1,I)
320  NEXT I: NEXT J: REM  SE BUSCAN
     OTROS POSIBLES RESULTADOS
330 M2 = PD(MJ):MY = MJ: FOR J = 1 TO
     Q
340  IF PD(J) > M2 THEN M2 = PD(J):M
     Y = J
350  IF PD(J) < D(MJ) THEN QC(J) = 0
     : PRINT "NO PUEDE SER ";Q$(J): REM
      ESTE RESULTADO YA NO ES POSIBLE

360  NEXT J
370  IF PD(MY) > PD(MJ) THEN  GOTO 1
     70: REM  TODAVIA HAY INCERTIDUMB
     RE POR LO QUE SE NECESITA OTRA V
     ARIABLE
380  PRINT "ES ";Q$(MY);" EL RESULTA
     DO? ";: INPUT A$: IF A$ = "S" THEN
      PRINT "ALBRICIAS!": PRINT "PULS
     E CUALQUIER TECLA PARA SEGUIR": GET
     X$: GOTO 150
390  FOR I = 1 TO Q: PRINT I;" ";Q$(
     I): NEXT I: INPUT "CUAL ES EL RE
     SULTADO? ";P
400  FOR J = 1 TO V: IF VC(J) = 1 THEN
      PRINT "QUE VALOR TENIA LA VARIA
```

```
    BLE ";J;" (";V$(J);") ? ";: INPUT
    V(J):VC(J) = 0
410  NEXT J
420  FOR I = 1 TO Q: IF D(I) >  = D(
    P) AND I <  > P THEN  FOR J = 1 TO
    V:R(I,J) = R(I,J) - V(J): NEXT J

430  NEXT I
440  FOR J = 1 TO V:R(J,P) = R(J,P) +
    V(J): NEXT J
450  PRINT "LA MATRIZ DE REGLAS ACTU
    AL ES:"

460  PRINT  SPC( 10);: FOR J = 1 TO
    Q: PRINT Q$(J); SPC( 10 -  LEN (
    Q$(J)));: NEXT J: PRINT
470  FOR I = 1 TO V: PRINT V$(I); TAB(
    10);: FOR J = 1 TO Q: PRINT R(I,
    J); TAB( 10 + J * 20);: NEXT J: PRINT
    : NEXT I
480  PRINT "PULSE CUALQUIER TECLA PA
    RA SEGUIR": GET X$: GOTO 150
```

Listado para el Sinclair Spectrum

```
   2  REM Observe que a los arrays se les
ha cambiado el nombre, porque en el Spectr
um solo pueden tener una letra.
  10 CLS
  20 INPUT "Cuantas variables? ";v
  25 DIM v(v): DIM v$(v,10): DIM m(2,v): D
IM c(v): DIM p(v): DIM a(v)
  30 PRINT AT 16,0;"Por favor, nombralas:"
  40 FOR i=1 TO v: INPUT "La variable ";(i
);" es ";v$(i)
  45 INPUT "Tiene un valor minimo=";m(1,i)
: INPUT "Y un valor maximo=";m(2,i)
  50 NEXT i
  55 CLS
  70 INPUT "Cuantos resultados tienes? ";q
  75 DIM q$(q,20): DIM r(v,q): DIM d(q): D
IM q(q)
  80 PRINT AT 16,0;"Por favor, nombralos"
  90 FOR i=1 TO q: INPUT "El resultado ";(
i);" es ";q$(i): NEXT i
 100 CLS : PRINT "Variable";TAB 12;"Minimo
";TAB 22;"Maximo": PRINT : FOR i=1 TO v: P
RINT v$(i);TAB 12;m(1,i);TAB 22;m(2,i): NE
XT i: PRINT : PRINT "Pulsa cualquier tecla
 para continuar"
 105 LET a$=INKEY$: IF a$="" THEN  GO TO 1
05
 110 CLS
 120 PRINT "Esto es una sesion de entrenam
iento": PRINT "Por favor, da valores de ":
 PRINT "variables": PRINT : PRINT "Selecci
onare un resultado": PRINT : PRINT "Dime s
i es correcto"
 125 PRINT : PRINT "Pulsa cualquier tecla
para continuar"
 126 LET a$=INKEY$: IF a$="" THEN  GO TO 1
26
 128 CLS : PRINT "Dime tus variables"
 130 FOR i=1 TO v: LET c(i)=1: LET v(i)=0:
```

```
 NEXT i: FOR j=1 TO q: LET d(j)=0: LET p(j
)=0: LET q(j)=1: NEXT j
 160 LET vr=0: LET vv=1: FOR i=1 TO v: LET
 a(i)=0: FOR j=1 TO q: LET a(i)=a(i)+ABS (
m(1,i)-m(2,i))*ABS (r(i,j))*c(i)*q(j): NEX
T j: IF a(i)>vr THEN  LET vv=i: LET vr=a(i
)
 180 NEXT i: REM Eligio un valor prometedo
r
 185 IF vr>0 THEN  GO TO 200
 190 PRINT : PRINT "En este momento, tiene
s"'"que decirme el resultado"''
 192 PRINT "Pulsa cualquier tecla para con
tinuar"
 194 LET a$=INKEY$: IF a$="" THEN  GO TO 1
94
 196 GO TO 390
 200 INPUT "El numero ";(vv);" (";(v$(vv))
;") es ";v(vv)
 210 LET c(vv)=0: REM Pone el flag a cero
 220 FOR j=1 TO q
 230 LET d(j)=d(j)+v(vv)*r(vv,j)
 240 NEXT j: REM Actualiza el vector de de
cision
 250  LET d=d(1): FOR j=1 TO q
 260 IF d(j)>=d THEN  LET d=d(j): LET hj=j
 270 LET p(j)=d(j): NEXT j: REM Se encuent
ra la mejor conjetura hasta ahora
 280 FOR j=1 TO q
 290 FOR i=1 TO v
 300 IF c(i)=1 THEN : IF r(i,j)>r(i,mj) TH
EN  LET p(j)=p(j)+(r(i,j)-r(i,mj))*m(2,i)
 310 IF c(i)=1 THEN : IF r(i,j)<r(i,mj) TH
EN  LET p(j)=p(j)+(r(i,j)-r(i,mj))*m(1,i)
 320 NEXT i: NEXT j: REM Se obtienen todas
 las alternativas posibles
 330 LET h2=p(mj): LET mi=mj: FOR j=1 TO q
 340 IF p(j)>h2 THEN  LET h2=p(j): LET mi=
j

 350 IF p(j)<d(mj) THEN  LET q(j)=0: PRINT
```

```
 "No puede ser ";(q$(j)): REM Ya no es pos
ible
 360 NEXT j
 370 IF p(mi)>p(mj) THEN  GO TO 160: REM T
oma otra variable
 380 INPUT "Es el resultado ";(q$(mi));"?"
;a$: IF a$="s" THEN  PRINT '''"BRAVO!"
 381 IF a$<>"s" THEN  GO TO 390
 382 FOR i=1 TO 100: NEXT i: PRINT "Pulsa
cualquier tecla para continuar"
 385 LET a$=INKEY$: IF a$="" THEN  GO TO 3
85
 386 GO TO 128
 390 CLS : FOR i=1 TO q: PRINT i;" ";(q$(i
)): NEXT i: INPUT "Que resultado es?";p
 400 CLS : FOR j=1 TO v: IF c(j)=1 THEN  P
RINT AT 16,0;"Que  valor tenia"'"la variab
le ";(j);"(";(v$(j));") ?": INPUT v(j): LE
T c(j)=0
 410 NEXT j
 420 FOR i=1 TO q: IF d(i)>=d(p) AND i<>p
THEN : FOR j=1 TO v: LET r(j,i)=r(j,i)-v(j
): NEXT j
 430 NEXT i
 440 FOR j=1 TO v: LET r(j,p)=r(j,p)+v(j):
 NEXT j
 450 CLS : PRINT "La matriz de reglas actu
al es como sigue, para cada resultado"
 460 PRINT : PRINT
 470 FOR j=1 TO q
 480 PRINT q$(j): PRINT
 490 FOR i=1 TO v
 500 PRINT v$(i),r(i,j)
 510 NEXT i
 520 PRINT "Pulsa cualquier tecla para con
tinuar"
 530 LET a$=INKEY$: IF a$="" THEN  GO TO 5
30
540 CLS : NEXT j
550 GO TO 128
```

5. UN EXPERTO EN EL MUNDO REAL

5.1 DE NUEVO, EL TIEMPO

Está muy bien disponer de un sistema experto que puede contestar a preguntas ingeniosas sobre pájaros, aviones y planeadores. Resulta un bonito juguete. Pero, ¿qué pasa si le planteamos un problema real, tal como el del tiempo?. ¿Qué hará el sistema si le preguntamos si va a llover mañana?.

Bien, desde luego, es un problema real. Nosotros mismos no sabemos muy bien responder a esa pregunta, de modo que, si el experto pudiera hacerlo, nos convenceríamos de que hemos conseguido algo útil.

Por tanto, dejemos los ejemplos artificiales y vayamos al Centro Meteorológico de Londres para obtener algunos datos reales.

El Centro Meteorológico de Londres publica mensualmente las cifras diarias de la precipitación, temperatura y horas de insolación. Fijándonos en un día cualquiera, podemos proporcionarle datos al experto y entrenarlo, haciéndole saber si al día siguiente llovió o no.

Conforme progresemos a lo largo de los registros de datos, veremos (quizás) que las opiniones de nuestro experto van mejorando.

Las cifras de la Tabla 5.1 corresponden a marzo de 1982, y dan las temperaturas máximas y mínimas (en grados centígrados), la precipitación (en litros) y las horas de sol. Si la precipitación es nula, se trata (puede usted creerlo) de un día seco. En otro caso, consideraremos que está lloviendo.

Día del mes	Temp. Mín. °.C	Temp. Máx. °.C	Precip. litros por m²	Horas de sol
1	9.4	11.0	17.5	3.2
2	4.2	12.5	4.1	6.2
3	7.6	11.2	7.7	1.1
4	5.7	10.5	1.8	4.3
5	3.0	12.0		9.5
6	4.4	9.6		3.5
7	4.8	9.4		10.1
8	1.8	9.2	5.5	7.8
9	2.4	10.2	4.8	4.1
10	5.5	12.7	4.2	3.8
11	3.7	10.9	4.4	9.2
12	5.9	10.0	4.8	7.1
13	3.0	11.9	0.2	8.3
14	5.4	12.1		1.8
15	8.8	9.1	8.8	
16	2.4	8.5	3.0	3.1
17	4.3	10.8	4.2	4.3
18	3.4	11.1		6.6
19	4.4	8.4	5.4	0.7
20	5.1	7.9	3.0	0.1
21	4.4	7.3	1.0	
22	5.6	14.0		6.8
23	5.7	14.0		8.8
24	2.9	13.9		9.5
25	5.8	16.4		10.3
26	3.9	17.0		9.9
27	3.8	18.3		8.3
28	5.8	15.4		7.0
29	6.7	8.8	6.4	4.2
30	4.5	9.6		8.8
31	4.6	9.6	3.2	4.2

Tabla 5.1
Resumen del tiempo en Londres durante marzo de 1982
(Los espacios en blanco indican ceros)

Vamos a dar unos valores máximos y mínimos, aunque no sería demasiado difícil que algún día las variables excedieran esos límites:

Temperatura:	mínimo 0° C	máximo 20° C
Precipitación:	mínimo 0 litros	máximo 25 litros
Insolación:	mínimo 0 horas	máximo 12 horas

Aparte de los valores mínimos para precipitación y horas de sol, estas cifras podrían ampliarse para tener en cuenta el tiempo durante todo el año.

A propósito, fue el mes de marzo más soleado desde 1967, y el segundo desde que comienzan los registros, en 1929, de acuerdo con el Centro Meteorológico, donde son unos bien conocidos optimistas, que incluso, en su optimismo, llegan a decirnos que ayer hizo buen tiempo.

En cualquier caso, inicializaremos al sistema experto con cuatro variables: temperatura mínima, temperatura máxima, precipitación e insolación. Le damos lo que estimamos como valores máximos y mínimos razonables, y le damos dos resultados: Lloverá y No Lloverá.

El primer elemento corresponde al 1.° de marzo de 1982. Introducimos las variables y le hacemos saber al experto que al día siguiente llovió.

Estamos en el día siguiente. Le damos los valores del 2 de marzo, y, como el experto se equivoca sobre el tiempo que hizo al día siguiente, le decimos que también llovió.

Al tercer día, ¡el experto acierta!. Pero el entusiasmo no es bueno para este proceso. Porque, si echa usted un vistazo a la Tabla, verá que en marzo de 1982 hubo 18 días de lluvia y 13 secos. Lo cual sugiere que, aproximadamente, hay un 50 por ciento de posibilidades de acertar por puro azar. ¿Progresa, pues, el sistema experto?. ¿Es mejor que el puro azar?.

En la Tabla 5.2 puede verse que el experto ha predicho correctamente lluvia, o su ausencia, en 22 de los 30 días (el primer día no cuenta, porque el experto no tenía aún reglas establecidas). Esto representa el 73 por ciento de éxitos, y, verdaderamente, parece bastante bueno. Posiblemente, mucho mejor de lo que habría podido predecir una persona. Pero analicemos los resultados con un poco más de detalle.

Lo primero que hay que hacer es moderar un poco el (posible) entusiasmo originado por estos resultados. Y lo haremos señalando que hay

una regla extraordinariamente sencilla con la que sólo se habrían cometido diez errores durante todo el mes. La regla es: el tiempo que hará mañana será el mismo (lluvioso o no lluvioso) que hace hoy. Si utilizamos esta regla, sólo nos equivocaremos cada vez que un día sin lluvia precede a uno lluvioso, y cada vez que uno lluvioso precede a uno sin lluvia. En general, utilizando esta regla podemos esperar acertar casi siempre, porque sabemos que el tiempo va por rachas, y que la tendencia es a tener una racha de días secos seguida por otra de días lluviosos.

Día del Mes	Tiempo Real al día siguiente	Tiempo Predicho por el experto
1	Lluvioso	No sabe
2	Lluvioso	Lluvioso
3	Lluvioso	Lluvioso
4	Seco	Lluvioso - Error
5	Seco	Lluvioso - Error
6	Seco	Seco
7	Lluvioso	Seco - Error
8	Lluvioso	Lluvioso
9	Lluvioso	Lluvioso
10	Lluvioso	Lluvioso
11	Lluvioso	Lluvioso
12	Lluvioso	Lluvioso
13	Seco	Seco
14	Lluvioso	Lluvioso
15	Lluvioso	Lluvioso
16	Lluvioso	Lluvioso
17	Seco	Lluvioso - Error
18	Lluvioso	Seco - Error
19	Lluvioso	Lluvioso
20	Lluvioso	Lluvioso
21	Seco	Lluvioso - Error
22	Seco	Seco
23	Seco	Seco
24	Seco	Seco
25	Seco	Seco
26	Seco	Seco
27	Seco	Seco
28	Lluvioso	Seco - Error
29	Seco	Lluvioso - Error
30	Lluvioso	Lluvioso
31	Lluvioso	Lluvioso

Tabla 5.2
Tiempo real y predicho durante marzo de 1982 en Londres

Así, siguiendo este razonamiento, si el experto no hubiese hecho más que ajustar su base de reglas para predecir para mañana el mismo tiempo que hace hoy, entonces sólo habría cometido diez errores. Ciertamente, es preferible cometer sólo ocho errores, lo cual demuestra que el asunto no es tan fácil. Pero si observa usted los resultados, verá que el

experto ha sido sorprendido en los cambios de tiempo. No siempre ha sido así, como en el caso del 13 de marzo, pero generalmente sí.

Lo que sucede es que, después del primer día, el experto hace, en este caso, predicciones de lluvia, que son correctas hasta el momento en que (por si no se ha dado usted cuenta) son falsas, debido a la presencia de un día sin lluvia. Este error hace que el experto ajuste sus reglas, hasta que por fin llegan a modificarse lo suficiente para predecir un día seco, el 6 de marzo.

En ese día, el tiempo vuelve a ponerse lluvioso, y el experto se equivoca. En ese momento, ajusta sus reglas por tercera vez, y estas reglas le dan un buen servicio hasta el día 17, cuando insospechadamente aparece un día seco. Las reglas sufren nuevos ajustes, y el día 22 el experto funciona lo bastante bien para predecir un día seco. Los días 28 y 29, al volver las lluvias, aparecen más errores.

Desde cierto punto de vista, nuestro experto ha funcionado mal, porque comete errores periódicamente. No es perfecto. Y lo que puede llamar la atención a algunos es el hecho de que, como ajusta su base de reglas cada vez que comete un error, no está utilizando el mismo conjunto de reglas en el curso de su funcionamiento, lo cual puede hacerle a uno pensar que, en cierto sentido, el experto está haciendo trampa. Después de todo, alguien puede pensar, ¿qué es lo que estamos evaluando aquí? En la mayoría de los juegos no se le permite a uno que cambie las reglas cada vez que pierde.

Esta alteración de las reglas es lo que hace que nuestro sistema experto sea difícil de analizar, si no es probándolo para ver lo que hace. Si tuviese un conjunto de reglas inmutables, sería posible decir más cosas sobre ellas, pero no las tiene, sólo tiene un método general para proporcionar una respuesta.

En cierto sentido, esto hace al sistema más humano. Después de todo, si a usted le preguntaran qué tiempo piensa que va a hacer mañana, probablemente tendría en cuenta el tiempo que ha estado haciendo recientemente. En este caso, únicamente se considera el tiempo que hace hoy. En general, si el tiempo está siendo seco, un pequeño chaparrón no le hará cambiar de opinión sobre la posibilidad de que mañana siga siendo seco, o, en cualquier caso, es poco probable que cambie de opinión. Usted enfocará el asunto de la predicción del tiempo con cierta «inclinación» mental, o predisposición, para interpretar los indicios de hoy en el contexto del tiempo reciente en general.

Suponga, por ejemplo, que quisiera usted predecir si va a nevar mañana. Ahora bien, si estuviésemos a mitad de enero y recientemente hu-

biese habido muchas nevadas y todo indicase hoy que va a nevar, entonces usted quedaría satisfecho augurando una nevada. Pero si estuviésemos a mitad de julio y se presentaran los mismo signos que en mitad de enero presagiaban nieve, usted estaría menos dispuesto a predecir una nevada. El hecho es que usted está utilizando reglas diferentes en cada momento.

Y, si realmente nevase en mitad de julio, usted ajustaría rápidamente sus reglas internas de predicción meteorológica y estaría mucho más dispuesto a considerar la posibilidad de que mañana vuelva a nevar, si siguen apareciendo los mismos signos. Y eso es lo que hace nuestro experto. Dicho esto, su comportamiento empieza a parecer mucho más razonable.

También parece razonable su comportamiento en cuanto a solicitar información. Recuerde que nuestro experto elige libremente las variables cuyos valores necesita, y es libre para hacer una predicción sin preguntar sobre todas las variables, si está seguro del resultado.

En este ejemplo, el sistema hace uso de esa libertad. Casi invariablemente, la pregunta que primero plantea es: ¿llueve?. Obtenida una cifra para la precipitación de hoy, normalmente toma una decisión sin comprobar ninguna otra variable, y, como hemos visto, generalmente acierta.

Lo que parece hacer el sistema es predecir lluvia cuando la precipitación de hoy es alta, y predecir un día seco cuando la precipitación es baja. Cuánto es alto y cuánto es bajo, varía. Y, en medio, hay una zona gris en la que el sistema no puede formarse una opinión. En esta zona gris, a veces solicita los valores de otras variables para añadir a la información sobre la precipitación, preguntando sobre temperatura e insolación. Muy rara vez pide los valores de todas las variables involucradas.

En conjunto, su comportamiento parece razonable. También razonable es el conjunto de reglas que ha desarrollado al final del mes. La matriz R el día 31 de marzo de 1982 es la siguiente:

	Mañana Lloverá	Mañana No Lloverá
Temperatura Mínima	—0.6999	0.6999
Temperatura Máxima	—2.5	2.5
Precipitación	4.1	—4.1
Insolación	4.6	—4.6

Como puede usted ver, para predecir lluvia mañana, el experto mira si la cifra de precipitación de hoy es alta y la de la temperatura baja. Cu-

riosamente, también mira si la cifra de insolación es alta, como un predictor de lluvia. Pero quizá esto no sea tan extraño. El tiempo frío, lluvioso y con sol es ciertamente «tiempo inestable», que es, quizá, lo que el experto está buscando. Después de todo, al final de marzo bien podemos esperar que comiencen a aparecer en abril algunos chaparrones con sol.

Por otra parte, para predecir un día seco mañana, el experto mira si hoy no llueve, la temperatura es cálida y está nublado. Quizá sea éste el tipo de tiempo tranquilo que no presagia lluvia. Resumiendo, ésos no son los elementos de información que nosotros utilizaríamos automáticamente para predecir el tiempo de mañana, pero quizás deberíamos hacerlo. Si nuestro sistema experto puede hacer buenas predicciones del tiempo de este modo, podría ser que verdaderamente tuviese ya alguna experiencia que supera en algunos aspectos a la nuestra. Posiblemente, habiendo aprendido el experto de nosotros inicialmente, podamos ahora nosotros aprender algo de él.

Pero, exactamente, ¿cómo de bien ha funcionado el sistema?

Realmente, podemos ser un poco más precisos sobre cómo de bien funciona, diciendo que no sólo acierta en un 73 por ciento de los casos (en este ejemplo), sino que la probabilidad de que estos aciertos sean debidos al azar es inferior a 0,025, es decir, menos de 25 posibilidades entre mil. ¿Y cómo, se preguntará usted, podemos decir tal cosa?.

Pues bien, se impone un poco de estadística. Decimos que ha habido 17 días lluviosos y 13 días secos en marzo, y que el experto ha predicho 19 días de lluvia y 11 días secos; podemos representar los resultados del siguiente modo:

		Predicción del experto		
		Lluvia	Seco	Total
Tiempo real	Lluvia	14	3	= 17
	Seco	5	8	= 13
	Total	19	11	= 30

En 14 días, se predijo lluvia y llovió. En 8 días, se predijo un día seco, y lo fue. En los ocho días restantes se cometieron errores.

Utilizando los totales, podemos calcular qué cifras se habrían obtenido por el puro azar. Estas cifras son los llamados «valores esperados», y al final de este Apartado se indica cómo se pueden calcular:

Valores esperados

		Lluvia	Seco	Total
Tiempo	Lluvia	10.77	6.23	= 17
real	Seco	8.23	4.77	= 13
	Total	19	11	= 30

Podemos utilizar esas dos tablas en una prueba estadística llamada la «prueba de ji-cuadrado».

La prueba de significación de ji-cuadrado dice que la tabla construida a partir de las predicciones del experto es diferente de la tabla basada en el azar, y que es tan diferente que la probabilidad de que el experto esté funcionando al azar es inferior a 0,025.

El problema con este tipo de afirmación es evidente: puede que usted no conozca lo que es una prueba de ji-cuadrado, y en tal caso, puede que se crea la afirmación o que no.

Pues bien, si lo prefiere, puede aceptarla y creérsela. Y si realmente quiere saber cómo funciona la prueba, lea lo siguiente.

La primera tabla contiene las frecuencias observadas, a las que llamaremos O (por «observadas»). La segunda tabla contiene las frecuencias esperadas, a las que llamaremos E (por «esperadas»).

El valor de ji-cuadrado se calcula así:

JI = JI + ((O-E) ^ 2)/E

sumando para cada una de las cuatro células de la tabla.

Es decir, en el caso de nuestro ejemplo:

JI = ((14-10,77) ^ 2)/10,77) + ((3-6,23) ^ 2)/6,23 + ((5-8,23) ^ 2)/8,23)
+ //8-4,77) ^ 2)/4,77).

con lo que se obtiene:

JI = 6,11

Ahora, busque usted el valor de ji-cuadrado en unas tablas de estadística (lo siento, tiene usted que comprar unas tablas de estadística). Busque los valores para un grado de libertad, y verá que en la columna correspondiente a 0,025 (algunos dicen 2,5 por ciento) ji-cuadrado tiene

un valor de 5,02. Nuestro valor de ji-cuadrado es mayor, por lo que podemos confiar, con una probabilidad de al menos 0,025, en que nuestros resultados no son debidos al azar.

Si encuentra usted todo esto algo tedioso, siempre puede prescindir de ello. Pero le permite juzgar la eficacia de su sistema experto de una forma muy exacta. Después de todo, usted puede tener otro problema diferente, y puede que quiera estimar hasta qué punto su experto está funcionando bien.

Si está usted interesado, puede que se pregunte qué son los grados de libertad y cómo se obtienen los valores esperados.

Los grados de libertad dependen del número de filas y de columnas que tiene la tabla, y se calculan así:

GL = (F-1)*(C-1), donde F es el número de filas, C el de columnas, y GL son los grados de libertad.

En este caso, F = 2, C = 2, por lo que GL = 1. Pero podría usted tener más resultados, y, por tanto, una tabla mayor.

Los valores esperados se calculan para cada celda del siguiente modo:

Mirar el total (observado) de la fila correspondiente a esa celda
Mirar el total (observado) de la columna.
Multiplicar ambos números.
Dividir el resultado por el total general (observado).
Repetir este proceso para todas las celdas.
Introducir los valores en la tabla de Valores Esperados.

Si no se equivoca usted en los cálculos, la tabla Esperada debe tener los mismos totales de filas, de columnas y general que la tabla Observada.

En nuestro ejemplo, la primera celda de la tabla esperada es 10,77, porque 17*15/30 = 10,77.

Realmente, es un asunto ideal para un programa de ordenador, si está usted interesado todavía y aún no se ha dormido.

Si no tiene a mano tablas estadísticas, observe simplemente que si el valor de ji-cuadrado es cero, entonces los resultados ciertamente podrían ser debidos al azar, y que, cuanto mayor es ji-cuadrado, tanto mejor para nuestros propósitos.

5.2 UN PROGRAMA DE JI-CUADRADO

Si le interesa comprobar los resultados de los expertos de un modo formal, utilizando la prueba de ji-cuadrado, y no quiere aplicar a mano el método anterior, puede tener un programa que se lo haga.

Puede apuntar sus resultados, como antes, en una tabla, y luego ejecutar el siguiente programa en Applesoft:

```
10  HOME : INPUT "CUANTAS FILAS HAY?
     ";F
20  INPUT "CUANTAS COLUMNAS HAY? ";C

30  DIM Q(F + 1,C + 1),E(F,C)
40  FOR I = 1 TO F
50  FOR J = 1 TO C
60  PRINT "FILA ";I;" COLUMNA ";J;"
    = ";: INPUT Q(I,J)
70  NEXT J: NEXT I
80  FOR I = 1 TO F
90  FOR J = 1 TO C
100 Q(F + 1,J) = Q(F + 1,J) + Q(I,J)

110 Q(I,C + 1) = Q(I,C + 1) + Q(I,J)

120 Q(F + 1,C + 1) = Q(F + 1,C + 1) +
    Q(I,J)
130  NEXT J: NEXT I
140  FOR I = 1 TO F
150  FOR J = 1 TO C
160 E(I,J) = Q(I,C + 1) * Q(F + 1,J)
     / Q(F + 1,C + 1)
170  NEXT J: NEXT I
180  FOR I = 1 TO F
190  FOR J = 1 TO C
200 JI = JI + ((Q(I,J) - E(I,J)) ^ 2
    ) / E(I,J)
210  NEXT J: NEXT I
220 GL = (F - 1) * (C - 1)
230  PRINT "JI CUADRADO = ";JI: PRINT
    "GRADOS DE LIBERTAD = ";GL
```

El programa le da el valor de ji-cuadrado. Y, lo siento, pero sigue necesitando tablas.

El programa equivalente para el Sinclair Spectrum es el siguiente:

```
 10 CLS : INPUT "Cuantas filas? ";r
 20 INPUT "Cuantas columnas? ";c
 30 DIM q(r+1,c+1): DIM e(r,c)
 40 FOR i=1 TO r
 50 FOR j=1 TO c
 60 PRINT AT 16,0;"Fila ";i;" Columna ";j
;" es?"
 65 INPUT q(i,j)
 70 NEXT j: NEXT i
 80 FOR i=1 TO r
 90 FOR j=1 TO c
100 LET q(r+1,j)=q(r+1,j)+q(i,j)
110 LET q(i,c+1)=q(i,c+1)+q(i,j)
120  LET q(r+1,c+1)=q(r+1,c+1)+q(i,j)
130 NEXT j: NEXT i
140 FOR i=1 TO r
150 FOR j=1 TO c
160 LET e(i,j)=q(i,c+1)*q(r+1,j)/q(r+1,c+
1)
170 NEXT j: NEXT i
175 LET ch=0
180 FOR i=1 TO r
190 FOR j=1 TO c
200 LET ch=ch+(ABS (q(i,j)-e(i,j))^2)/e(i
,j)
210 NEXT j: NEXT i
220 LET df=(r-1)*(c-1)
230 CLS : PRINT AT 20,0;"Chi Cuadrado= ";
ch'"Grados de libertad=";df
```

5.3 CÓMO ENTRENAR A SU EXPERTO

En este momento, hay una observación que, para ser francos, hemos de hacer sobre nuestro sistema experto.

Desde muchos puntos de vista, es excelente. No hay nada malo en él. Funciona. Adquiere su experiencia con poca ayuda por nuestra parte. Sólo necesita que se le den unos cuantos ejemplos para empezar.

Pero, ¿cuántos ejemplos necesita?. ¿En qué momento podrá empezar a hacer juicios expertos de la mejor calidad?.

Pues bien...Considerando el ejemplo del pájaro/avión/planeador, podría usted suponer, inocentemente, que tres ejemplos bastarían: uno de cada tipo. Después de todo, una vez que ha visto una cosa cada vez, ha visto las tres. Cierto, las habrá visto todas. Pero no es cierto que haya desarrollado reglas adecuadas para volver a reconocerlas. El conseguir un método de trabajo realmente bueno, le llevará más de tres intentos manipulando su conjunto de reglas.

Entonces, para ser precisos, ¿cuántos ejemplos necesita exactamente el experto que se le den, en números reales?.

Bueno (en este momento uno se siente cogido y trata de eludir, por así decirlo, la mirada del lector), la teoría, en realidad, cuenta con que el experto tenga acceso a cada uno de los posibles ejemplos de entrada, un número infinito de veces, para ser precisos. Eso es lo que dice la teoría.

Pero la buena noticia es que, si los resultados son linealmente separables, el experto desarrollará en un tiempo finito un conjunto de reglas que le permitirá identificar todos los resultados posibles.

Así, con el ejemplo del pájaro/avión/planeador, podría ser que estuviese usted dispuesto a darle al experto cada una de las tres posibilidades un número infinito de veces, pero si existe un buen conjunto de reglas, el experto las encontrará en un tiempo finito. Por tanto, he aquí la respuesta a su pregunta: El sistema necesita basarse en 3*infinitos ejemplos para poder funcionar. Pero muy bien puede llegar a una conclusión aceptable antes de que los haya recorrido todos. (Y muy difícilmente se resiste uno a sugerir que podría necesitar sólo la mitad de esos 3*infinitos ejemplos).

En este momento, habrá, sin duda, lectores que habrán echado a un lado este librito, ya sea por considerarlo intrínsecamente inútil, o pensando en que, si es preciso recopilar una colección casi infinita de datos, mejor es empezar ahora mismo.

Sin embargo, le recomiendo aguardar un momento. Vamos a reducir un poco el problema.

En primer lugar, en teoría podría haber un rango infinito de entra-

das diferentes a dar. Pero, en la práctica, usted no va a tener registradas todas esas posibilidades (ocuparían demasiado espacio...).

En la práctica, usted tendrá una bonita pila de datos, finita, para que el experto comience a funcionar, y, si usted le echa todos los ejemplos que tiene, su funcionamiento debe llegar a ser razonablemente bueno en un tiempo finito.

De modo que el problema realmente se reduce a cómo darle al experto suficientes ejercicios para la resolución de problemas, sin que tenga usted que pasarse el resto de su vida al teclado, y sin que tenga que irse todas las noches a la cama con llagas en los dedos.

Afortunadamente, ello resulta muy fácil. Sólo es un asunto de estúpida repetición, que es para lo que los ordenadores se construyen originalmente. Todo lo que usted tiene que hacer es coger todos sus ejemplos, ponerlos en una matriz, por ejemplo, E(V + 1,N) y escribir una pequeña rutina para que el experto se ejercite por sí mismo durante su ausencia.

La idea es que usted ponga los ejemplos, deje al experto trabajando en ellos, y luego se vaya a un bar, donde puede decirle a la concurrencia que su propio sistema experto en este momento está haciendo todo el trabajo, y que, por consiguiente, usted no tiene nada que hacer. Ese tipo de comentario siempre ayuda a ganarse el respeto de la audiencia, porque, en su mayoría, ellos no saben si usted está diciendo la verdad o no, pero, si se lo creen, se caerán de la impresión.

Por tanto, reprimiendo el impulso de irse antes de dejar al experto trabajando, procedamos:

En primer lugar, E(V + 1,N). Obviamente, N es el número de ejemplos que vamos a darle al experto. Igualmente obvio es que la primera dimension sirve para guardar los valores de las V variables. El elemento V + 1 se utiliza para guardar el resultado de ese ejemplo.

En este momento (es decir, antes de empezar el proceso), puede hacer una pequeña trampa y, en vez de proporcionar estos datos desde el principio, rellenar la matriz E conforme introduce nuevos elementos al sistema experto en una sesión de entrenamiento. No importa cómo se haga; es usted quien debe decidir.

Todo lo que tiene que hacer ahora es poner un bucle de repetición sobre la rutina, de modo que el sistema trabaje repetidamente sobre esos ejemplos y vaya alterando sus reglas conforme lo hace. Un método po-

sible es utilizar el índice del bucle, por ejemplo, I, para seleccionar el siguiente elemento con el que debe trabajar el experto:

```
10  FOR I=1 TO N
20  FOR J=1 TO V
30  V(J)=E(J,I)
40  NEXT:NEXT
```

Este método tiene el inconveniente de que los ejemplos se dan siempre en el mismo orden. Quizá esto no importe en algunos casos. Pero suponga que tiene usted un conjunto de reglas que proporcionan un juicio correcto en el ejemplo I, y que también dan un juicio correcto en el ejemplo I + 1, por lo que no se cambian. Suponga ahora que el experto se equivoca en el ejemplo I + 2. Las reglas cambian, y quizá permiten obener la respuesta correcta en el ejemplo I + 3, pero, si a continuación viniese el ejemplo I, pudiera ser que las nuevas reglas no fuesen tan buenas para él, que dieran un error y que, en consecuencia, hubieran de modificarse de nuevo.

Evidentemente, para tratar de ser infalible, el experto debe ensayar todas las posibles órdenes de elección de los ejemplos, y trabajar sobre ellos una y otra vez. Si tenemos N ejemplos, hay N formas de elegir el primero, N-1 de elegir el segundo, N-2 de elegir el tercero. En general, hay N! (N factorial) formas de elegir los ejemplos, siendo N! = N*(N-1)*(N-2)*...*2*1.

Lo cual no sólo es una gran cantidad, también es difícil de programar en BASIC. Y, por añadidura, hay que hacerlo unas cuantas veces.

Un enfoque más razonable, que conduce a un programa más fácil y sirve para cualquier valor de N, consiste en seleccionar los ejemplos al azar, y hacerlo muchas veces. Supongamos que lo hacemos 100 veces:

```
50  FOR P=1 TO 100
60  I=INT(RND(1)*N+1)
70  NEXT
```

Sin duda, cualquiera puede sentirse satisfecho con ese trozo de programa, pero RND e INT no se interpretan del mismo modo en todas las máquinas, por lo que vamos a definir lo que hacen en el Apple:

La función RND(1) da, teniendo 1 como variable, un número aleatorio mayor o igual que cero y menor que 1. Por tanto, multiplicándola por N dará un número mayor o igual que cero y menor que N. Y, sumando 1 al resultado, se obtiene un número mayor o igual que 1 y que puede ser igual a N o un poco más, pero menor que N + 1.

La función INT da un número entero menor o igual a su variable, de modo que, en total, tenemos un número mayor o igual que cero y menor o igual que N, y que es entero.

Por tanto, podemos usar ese número para escoger un ejemplo.

De este modo, el experto irá haciendo aleatoriamente los deberes que usted le ha puesto, y, si quiere darle mucho tiempo, puede ponerlo en un bucle sin fin y dejarle que siga indefinidamente.

Teniendo en cuenta nuestros profundos estudios previos, el resto del programa es francamente fácil:

```
 80  FOR P=1 TO N: D=0: C=0
 90  I=INT(RND(1)*N+1)
100  Q1=E(V+1,I): REM: Q1 SERIA EL RESULTADO REAL
     CON ESTOS DATOS
110  FOR J=1 TO V
120  D=D+E(J,I)*R(J,Q1)
130  NEXT: REM: CON ELLO SE HALLA EL VALOR DE LA
     REGLA DE DECISION CON EL RESULTADO CORRECTO
140  FOR K=1 TO Q
150  D2=0
160  IF K<>Q1 THEN: FOR J=1 TO V: D2=D2+E(J,I)*(J,K):
     NEXT: REM: CON ELLO SE COMPRUEBA EL VALOR DE
     LA REGLA DE DECISION CON LOS RESULTADOS FALSOS
170  IF D2>=D THEN: FOR J=1 TO V: R(J,K)=R(J,K)-E(J,I):
     NEXT
180  IF D2>=D AND C=O THEN: FOR J=1 TO V: R(J,Q1)=R
     (J,Q1)+E(J,I): NEXT: C=1: REM: CON ELLO SE AJUSTAN
     LAS REGLAS SI LAS REGLAS ACTUALES HUBIESEN
     DADO UNA RESPUESTA INCORRECTA. C ES UNA SEÑAL
     PARA ASEGURARSE DE QUE Q1 SOLO SE AJUSTA UNA
     VEZ
190  NEXT K: NEXT P
```

Ahora, naturalmente, usted puede irse al bar que más le guste, dejando al experto con sus deberes. Pero lleve consigo este librito (siempre causa buena impresión parecer capaz de leer un libro), porque todavía hay algo más que decir.

Cuando regrese usted, ¿habrá desarrollado el experto el mismo conjunto de reglas que si usted hubiese trabajado sin descanso al teclado durante el resto de su vida?.

Podría usted pensar que sí. Pero podría ocurrir que no. Porque los datos se le están dado al experto de un modo diferente.

Observe. El experto sólo modifica sus reglas cuando comete un error. Y sólo emite un juicio cuando está totalmente seguro de que no pueden introducirse nuevos valores de variables que le hagan cambiar de opinión, dados los valores mínimos y máximos que tan amablemente le ha proporcionado usted.

Por tanto, al teclado, podría ocurrir que decidiese que nada puede hacerle cambiar de opinión, que emitiese un juicio, y que fuese correcto, y, por consiguiente, no cambiaría las reglas. Supongamos ahora que dos reglas, para un determinado ejemplo, dieran el mismo valor: se trata de un empate. Nuestro programa del Capítulo anterior no modifica las reglas a menos que el resultado que aventura, en caso de empate, sea el falso. Si puede escaparse sin modificar las reglas, lo hace, porque esto le evita a usted la molestia de darle más información.

Pero en este caso, usted no tiene que molestarse ante el teclado, por lo que el programa anterior comprueba todas las posibilidades y modifica las reglas en caso de empate, hasta que no se dan más empates. Por lo tanto, modificará sus reglas más a menudo.

Por ejemplo: Suponga que tiene usted un pájaro. El sistema comprueba sus reglas actuales y descubre que podría ser un pájaro, pero que, igualmente, podría ser un planeador. Pero el pájaro es el primer elemento que encuentra en la lista, y como el planeador sólo da un resultado igual, el experto dice que se trata de un pájaro. Correcto, le responde usted. Y las reglas permanecen sin modificarse, dando los mismos valores para pájaro y para planeador cuando la entrada es pájaro. El sistema, sin embargo, funciona.

Pero en el programa que acabamos de dar, el experto no va a admitir un conjunto de reglas con el que el pájaro y planeador den el mismo valor. Modificará las reglas hasta que los valores sean diferentes para todos los resultados.

Realmente, esto no significa que un conjunto de reglas sea mejor que otro. Después de todo, si da las respuestas correctas, ¿quién puede cuestionar su funcionamiento?

Si quisiera usted tener las mismas reglas en ambos casos, tendría que hacer un poco más complicado el programa para ejercitar a su experto. Por ejemplo, podría modificarlo para que las reglas sólo cambiaran:

IF D2>D (en vez de IF D2>=D)

o

IF D2 = D AND K>Q1

lo que significa que las reglas se modifican si se encuentra una regla que da un resultado incorrecto con una puntuación igual a la del resultado correcto, y este resultado incorrecto ocurre *antes* que el resultado correcto. Si ocurre después, no origina ningún problema, porque entonces se elegirá el resultado correcto, y cualquier otro que dé una puntuación igual a la suya no lo sustituirá.

Ello no es más que una pequeña diferencia, pero cuando ponga en funcionamiento a su sistema experto, observará algunos efectos prácticos. El más práctico consiste simplemente en que puede que le pida más información sobre las variables, antes de formarse una opinión. Esto es debido a que sigue pidiendo información hasta que está seguro de que no pueda ocurrir nada que le haga cambiar de opinión. Al hacer más cambios en las reglas que antes, los números reales que hay en los elementos de la matriz de reglas R(I,J) pueden ser un poco mayores o un poco menores. Por ello, habrá más posibilidades de variaciones entre los diferentes valores de los resultados. Y así, cuando el experto comprueba los valores mínimos y máximos de las variables que usted no ha introducido aún, le parecerá que hay más incertidumbre que en el otro caso. Y le hará más preguntas para tratar de despejar esta incertidumbre.

Si usted quiere ser purista, lo correcto es utilizar, para ejercitar al experto, los mismos métodos que normalmente se utilizan para trabajar con él. Esto implicaría o bien escribir prácticamente dos veces el mismo programa, o bien mezclar dos fuentes posibles de entradas (del teclado y de la matriz E(V + 1,N)) en un mismo trozo de programa. Cualquiera de los casos requiere un poco más de trabajo y de complicación, y, como es usted quien tiene que hacerlo, usted decidirá si quiere o no probarlo.

5.4 CÁLCULO DIRECTO

Lo que hemos hecho hasta aquí ha sido simular que nuestro experto sigue una sesión de entrenamiento, del mismo modo que si estuviésemos sentados ante el teclado dándole los ejemplos uno tras otro. Y no hay motivo por el que no podamos seguir así. Después de todo, funciona.

Pero lo esencial de la técnica es llegar a un conjunto de reglas que proporcionen juicios aceptables, y podríamos hacer algo mejor que de-

jar a la máquina manipular números sobre la base principal de un generador de números aleatorios.

Cuando considerábamos el caso de sólo dos resultados, señalábamos que es posible ajustar las reglas después de cada ejemplo, tanto si las reglas actuales dan la respuesta correcta como si no, y lo mismo podemos hacer en el caso de más de dos resultados.

Ahora, nos olvidaremos del elemento aleatorio y recorreremos todos los ejemplos que damos al sistema del siguiente modo:

```
10  FOR K=1 TO N (suponemos que tenemos N ejemplos)
20  FOR I=1 TO Q (para considerar todos los posibles resultados)
30  FOR J=1 TO V (para considerar todas las variables)
40  IF I=E(V+1,K) THEN: R(J,I)=R(J,I)+E(J,K)
50  IF I<>E(V+1,K) THEN: R(J,I)=R(J,I)—E(J,K)
60  NEXT: NEXT: NEXT
```

Lo que hacemos con ello es sumar los valores del ejemplo a la regla que da el resultado correcto, y restar los valores del ejemplo de todas las reglas que pertenecen a otros resultados. No se tiene en cuenta la cuestión de si las reglas actuales habrían funcionado o no.

Al final, después de recorrer todos los ejemplos de esta manera, nos quedamos con un conjunto de reglas cuyos valores conocemos (en teoría) con bastante precisión.

Supongamos, por ejemplo, que tenemos tres resultados. Piense en ellos como las esquinas de un triángulo suspendido en alguna parte del espacio, como en la Figura 5.1. Las reglas consisten en vectores (líneas que siguen direcciones particulares), cada uno de los cuales apunta a una esquina del triángulo y se aleja de las otras dos.

Las esquinas del triángulo vienen determinadas, de hecho, por los valores medios de las variables para cada uno de los resultados. Supongamos que el resultado 1 tiene el valor medio x_1 para ciertas variables, el resultado 2 tiene el valor medio x_2, y el resultado 3 el x_3. Entonces, el vector de regla para el resultado 1 tiene la dirección x_1—x_2—x_3.

Análogamente, el vector de regla para el resultado 2 tiene la dirección x_2—x_1—x_3, y así sucesivamente.

Cada vector de regla tiende a seguir una trayectoria, coincidente, en lo posible, con su propio valor medio. Y, al restar los valores medios de los otros resultados, también tiende a alejarse de ellos todo cuanto puede.

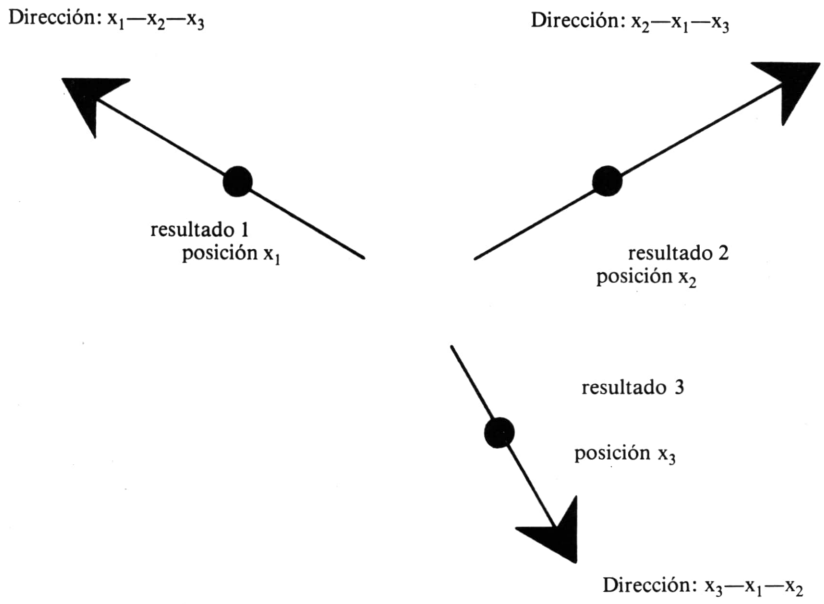


Fig. 5.1

Vectores de dirección para 3 resultados

Puede que usted haya observado que, realmente, no estamos utilizando valores medios. Si, por ejemplo, había n_1 ejemplos del resultado 1, no hemos dividido x_1 por n_1 para obtener el verdadero valor medio.

Considere en primer lugar lo que ocurriría si hubiese un número igual de ejemplos para cada resultado. En este caso, es indiferente dividir o no por el número de objetos. Como sólo importa su tamaño relativo, nada se gana ni se pierde.

Suponga ahora que hay un número diferente de ejemplos de cada clase. Bien, esto podría ser importante. Si le da usted al experto un conjunto representativo de ejemplos, y ese conjunto contiene, por ejemplo, un resultado que aparece dos veces más a menudo que cualquier otro, ésta es una información útil. El conocimiento de que un resultado es dos veces más probable que cualquier otro, y el hecho de que hay el doble de ejemplos de él, podría hacer que usted (o el experto) pensara que hay que creer (¡por utilizar un término científico!) el doble en ese resultado.

Por tanto, una vez obtenidos los valores medios para ese resultado, podríamos «ponderarlos», multiplicándolos por 2.

En general, si hay n_1 ejemplos de x_1, multiplicaríamos x_1 por n_1. Y si x_1 era el valor medio de los n_1 ejemplos, hemos vuelto al principio. Bastaría, simplemente, sumar todos los ejemplos.

La dificultad radica en que las cosas no son siempre tan fáciles como parecen. Si todavía estuviésemos tratando de predecir la diferencia entre un pájaro y un avión, y le diésemos al experto un ejemplo de cada uno, podría cogerlos y elaborar un conjunto de reglas. Pero suponga que, por alguna razón, le damos al experto dos ejemplos de lo mismo. ¿Qué significa esto?

El mismo conjunto de reglas seguiría separando bastante bien las diferentes posibilidades, y no hay razón para pensar que un resultado es doblemente probable que el otro. En definitiva, lo que usted le da al experto puede que no sea en absoluto representativo. Y eso origina un pequeño problema. Es usted, realmente, quien tiene que decidir qué hay que hacer aquí.

Para empezar, usted puede suponer que los valores que da son completamente representativos y van bien para hallar el valor medio de cada resultado. Después de eso, usted puede considerar aparte la cuestión de si el número de ejemplos para cada resultado es importante o no. Si lo es, puede ponderar los promedios por el número de ejemplos de cada resultado, y posiblemente obtenga algún beneficio. Pero si estima que el número de ejemplos de cada resultado no es una información verdaderamente útil (posiblemente, porque se los ha inventado sin reflexionar), entonces puede quedarse con los promedios ordinarios no ponderados.

Evidentemente, para obtener los promedios usted necesita llevar la cuenta de cuántos ejemplos ha dado de cada resultado posible. Esto puede hacerlo ocupando la memoria con otro vector, DIM N(Q), para llevar la cuenta. Así:

```
FOR I = 1 TO N
N(E(V + 1,I)) = N(E(V + 1,I)) + 1
NEXT
```

Y luego dividir los R(I,J) por los valores de N(J) para obtener los valores medios.

La única dificultad con este método es que, con toda franqueza, no siempre funciona. Considere los siguientes valores de V(I):

	Pájaro	Avión	Planeador
Alas	1	1	1
Pico	1	0	0
Motor	0	1	0

Es decir, un pájaro tiene alas y pico, un avión tiene alas y motor, y un planeador tiene alas y muy poco más. Ahora calcule un conjunto de reglas R(I,J) utilizando los valores medios:

	Pájaro	Avión	Planeador
Alas	—1	—1	—1
Pico	1	—1	—1
Motor	—1	1	—1

Finalmente, entregue a estas reglas, sucesivamente, cada uno de los conjuntos de variables, calculando los valores de D(J) para cada entrada V(I) a R(I,J):

Valores de D(J) para cada uno de los posibles resultados:

	Pájaro	Avión	Planeador
Pájaro	0	—2	—2
Avión	—2	0	—2
Planeador	—1	—1	—1

Evidentemente, no ha conseguido llegar a identificar al planeador, aunque puede averiguar sin dificultad los otros resultados.

Este problema no habría surgido si Planeador tuviera alguna otra variable que le identificase inequívocamente, como Cable de remolque, por ejemplo. Pero no la tiene, y el método ha fracasado. Puede arreglarse fácilmente contando Planeador dos veces, de modo que para Alas tengamos —2 con Pájaro y Avión, y 0 con Planeador, pero no sabemos tal cosa hasta que no revisamos las reglas para ver si han funcionado.

Por tanto, aunque el intentar un cálculo directo de las reglas podría parecer útil, siempre es una buena idea revisarlas posteriormente. Quizá simplemente haciendo funcionar al experto sobre unos pocos ejemplos. Pero, al hacerlo, también puede dejarse que el experto construya sus propias reglas por su cuenta, lo que nos conduce de nuevo al método de ejercitar al experto, como hacíamos antes.

6. FUNCIONAMIENTO DEL EXPERTO CON CASOS REALES

6.1 CÓMO UTILIZAR SU EXPERTO

En este momento tiene usted ya, lo crea o no, todo un sistema experto, hecho y derecho, de propósito general, con el que puede asombrar a sus amigos y colegas. Ha pasado usted por todas las etapas importantes, que, por si no se ha dado cuenta, han sido las siguientes:

Le ha dado al sistema los nombres de todas las posibles variables, y le ha comunicado los valores máximos y mínimos que cree usted que puede tomar cada variable. Las variables pueden ser intrínsecamente numéricas, o sólo categorías, como datos del tipo Sí/No. En cualquier caso, se tratan como entradas numéricas.

Le ha dado usted al sistema los nombres de todos los resultados posibles. Como el experto ha de elegir entre uno de esos resultados, conviene que se asegure usted de que siempre se puede aplicar al menos uno de ellos. Idealmente, deberían ser mutuamente exclusivos, y la suma de las probabilidades de que ocurra cada uno debería ser igual a la unidad. Lo cual significa que sólo puede darse un resultado, y que este resultado debe ser el que realmente ocurre.

Le ha dado, después, algunos ejemplos (tantos como ha podido) de entradas adecuadas, y le ha dicho cuál era el resultado para cada entrada.

Luego ha llevado a cabo una sesión de entrenamiento, durante la cual el experto no ha parado de barajar los ejemplos que le ha dado y de modificar su conjunto de reglas hasa obtener uno que le proporciona

siempre las respuestas correctas, o, al menos, hasta que queda totalmente mareado de intentarlo.

Pero ahora, lo que usted realmente quiere es utilizar el sistema experto para hacer algo. Evidentemente, puede continuar haciendo lo mismo que en la sesión de entrenamiento. Supongamos que lo ha entrenado usted para predecir el tiempo.

Al amanecer, acude usted a su experto, teclea unas variables, y obtiene una predicción. No hay ningún problema. Y luego, al día siguiente, hace lo mismo. Y al día siguiente. Pero, ¿y si la predicción es falsa? Pues bien, si el experto está funcionando como en una sesión de entrenamiento, lo primero que querrá saber es si su predicción anterior ha sido correcta, para ajustar sus reglas en caso negativo. Bien, en este caso no hay ningún inconveniente para satisfacer su curiosidad sobre el particular. Puede comunicarle usted el resultado para el último conjunto de datos que le ha dado, y luego pedirle una predicción del tiempo para hoy.

Pero podría ocurrir que, en su precipitación para poner en el equipaje el paraguas, el impermeable, las botas de lluvia y todo el equipo necesario para un verano inglés, olvidase usted decirle al experto lo que pasó con su última predicción, o, simplemente, que quisiera olvidarlo.

Del mismo modo, usted podría haber entrenado a su experto para realizar diagnósticos médicos importantes. Tras una sesión de entrenamiento, se encuentra usted, sentado en su consultorio, con una cola de pacientes al otro lado de la puerta.

Entra el primer paciente, conecta usted la máquina, teclea sus síntomas, y el experto comunica por la pantalla el resultado elegido.

«Lo siento, amigo» —sentencia usted—. «Parece que no es su día. Sólo le queda una semana de vida. ¡El siguiente!».

Y entonces tiene que permanecer sentado frente a su segundo paciente durante una semana, para poder decirle a su sistema experto si el resultado que ha predicho para el primero ha sido correcto. Este es el tipo de cosas que puede hacerle perder clientela, si previamente no toma usted medidas para hacer frente a la situación. Y, además, los pacientes pueden ser portadores de microbios, por lo que usted debe intentar que no permanezcan demasiado tiempo en la consulta.

En resumen, lo importante es que, aunque darle continuamente toda la realimentación posible al experto es una buena idea, llega un momento en que lo que quiere es dejar el entrenamiento y obtener juicios del ex-

perto. Y, en términos de programación, lo que esto significa es omitir la rutina de aprendizaje, y suponer que todas las respuestas del experto son correctas. Ya no le preguntará si ha acertado o no, ni ajustará más sus reglas.

En casos de ejemplos muy bien definidos, como el del pájaro/avión/planeador, en el que el experto puede llegar siempre a la respuesta correcta, no importa que se deje de dar realimentación. Pero, en casos más usuales, como la predicción meteorológica, en los que sabemos, o estamos casi seguros, que van a seguir apareciendo errores de vez en cuando, el experto va a perder algo al faltarle la realimentación. Pero, en tales casos, puede solventarse en gran parte el problema, guardando en memoria los detalles de los ejemplos que ocurren, y completando más adelante los resultados, cuando a uno le apetezca, o cuando estos resultados sean conocidos.

De ese modo, vamos recopilando un conjunto cada vez mayor de datos reales, que podemos utilizar periódicamente para ejercitar al experto, asegurándonos de que no pierde contacto con su tema.

Si volvemos ahora a nuestro ejemplo del tiempo, podrá usted comprobar que los resultados que se obtienen cuando se le hace trabajar en la realidad son diferentes de los que se obtienen durante una sesión de entrenamiento. Durante el entrenamiento, cada vez que se cometía un error, se ajustaban las reglas. Volvamos a probar ahora con los datos de marzo de 1982, utilizando las reglas obtenidas durante la sesión de entrenamiento, pero sin modificarlas. Los resultados son los de la Tabla 6.1.

Podríamos pensar, a primera vista, que los resultados no son extraordinariamente buenos: después de todo, ha cometido nueve errores en marzo. Pero si seguimos y aplicamos las mismas reglas, sin ninguna realimentación, al mes de abril (Tabla 6.2) nos encontramos tan sólo con seis errores en 29 días. Así, hemos obtenido, contando los dos meses, 15 errores en 60 días: una tasa de éxitos del 75 por ciento. Lo cual, de nuevo, no es perfecto, pero sí mejor que el puro azar. Después de todo, esa tasa de éxitos sugiere que, si realmente estuviera usted utilizando el experto, sólo se equivocaría 1,75 días en cada semana.

Y, para poner mejor las cosas, podemos considerar que, si el experto predice lluvia y ésta no llega, no importa demasiado; lo peor que puede suceder es que uno se pasee con un paraguas que no necesita. Lo malo es cuando el experto dice que no va a llover y llueve. Y esto sólo ha ocurrido en cuatro días de marzo y abril de 1982, aproximadamente un día de cada dos semanas. Por tanto, quizá los informes no sean tan malos.

Día del mes	Tiempo real al día siguiente	Tiempo predicho por el experto
1	Lluvioso	Lluvioso
2	Lluvioso	Lluvioso
3	Lluvioso	Lluvioso
4	Seco	Lluvioso - Error
5	Seco	Lluvioso - Error
6	Seco	Lluvioso - Error
7	Lluvioso	Lluvioso
8	Lluvioso	Lluvioso
9	Lluvioso	Lluvioso
10	Lluvioso	Lluvioso
11	Lluvioso	Lluvioso
12	Lluvioso	Lluvioso
13	Seco	Lluvioso - Error
14	Lluvioso	Seco - Error
15	Lluvioso	Lluvioso
16	Lluvioso	Lluvioso
17	Seco	Lluvioso - Error
18	Lluvioso	Lluvioso
19	Lluvioso	Lluvioso
20	Lluvioso	Lluvioso
21	Seco	Lluvioso - Error
22	Seco	Seco
23	Seco	Seco
24	Seco	Seco
25	Seco	Seco
26	Seco	Seco
27	Seco	Seco
28	Lluvioso	Seco - Error
29	Seco	Lluvioso - Error
30	Lluvioso	Lluvioso
31	Lluvioso	Lluvioso

Tabla 6.1
Pronósticos meteorológicos para marzo de 1982 en Londres

Día del mes	Tiempo real al día siguiente	Tiempo predicho por el experto
1	Lluvioso	Lluvioso
2	Seco	Seco
3	Seco	Seco
4	Lluvioso	Seco - Error
5	Lluvioso	Lluvioso
6	Lluvioso	Lluvioso
7	Lluvioso	Lluvioso
8	Lluvioso	Lluvioso
9	Seco	Lluvioso - Error
10	Seco	Seco
11	Seco	Lluvioso - Error
12	Seco	Lluvioso - Error
13	Seco	Lluvioso - Error
14	Seco	Seco
15	Seco	Seco
16	Seco	Seco
17	Seco	Seco
18	Seco	Seco
19	Seco	Seco
20	Seco	Seco
21	Seco	Seco
22	Seco	Seco
23	Seco	Seco
24	Seco	Seco
25	Seco	Seco
26	Seco	Seco
27	Seco	Seco
28	Lluvioso	Seco - Error
29	Seco	Seco

Tabla 6.2
Pronósticos del tiempo para abril de 1982 en Londres

Sin embargo, el punto interesante a observar sobre el tiempo de abril es que el experto no había sido entrenado sobre este tiempo. Supongamos que la predicción meteorológica fuese un problema linealmente separable. No lo es, y por ello se trata de una prueba tan difícil para nuestro sistema experto. Pero supongamos que lo fuese, y que el experto hubiese sido entrenado en el tiempo de marzo. Entonces, después de un entrenamiento adecuado, el experto podría predecir infaliblemente el tiempo para marzo. Pero ello no significaría que pudiese predecir el tiempo en abril, que podría tener valores completamente diferentes. El hecho de que haya producido un conjunto de predicciones bastante aceptable, es algo así como un premio, en cuanto que sugiere que hay algo acerca de la predicción meteorológica en general que nuestro sistema experto ha podido averiguar. Puede tomar datos completamente nuevos y obtener algo que parece una solución razonable, aunque ni nosotros mismos sabemos exactamente cómo predecir el tiempo, y lo único que hemos podido darle al experto ha sido el más general de los enfoques para trabajar y unos pocos ejemplos para ayudarle.

6.2 JUICIO RESERVADO

Si nuestro sistema experto, al intentar tomar una decisión, no pudiera formarse una opinión sobre el resultado, lo mejor que podría hacer sería decirlo. Admitir la derrota, antes que publicar una conjetura tontamente.

Desgraciadamente, esto es algo difícil para él.

Retrocedamos al ejemplo del pájaro/avión/planeador y supongamos que al principio de su entrenamiento el sistema se encuentra con que los tres resultados dan idénticos valores para una determinada entrada. Tal como lo hemos construido, el experto aventuraría un resultado, y, si fuese falso, modificaría sus reglas. Pero ahora decimos que, si todas las reglas diesen idénticos valores, debería responder «No Sé», y no pretender que tiene un conocimiento cuando realmente no lo tiene. Sí, muy bonito. Naturalmente, usted puede programarlo. Una pequeña modificación en el código, y, si hay más de un resultado posible, que imprima «No Sé», o, mejor, que liste los diferentes resultados posibles. Ello le dirá a usted muchas más cosas sobre el juicio que se hace, y puede incluirlo ya en el programa.

Pero, ¿el programa va a utilizar alguna vez este trozo adicional de código?. Sí, a veces. Así, con el ejemplo del pájaro/avión/planeador, el sistema puede dar una respuesta «No Sé» (juicio reservado) durante su entrenamiento. Pero, una vez que sus reglas han quedado establecidas,

ya no se reservará nunca el juicio, porque siempre obtiene la respuesta correcta. Lo cual es justamente lo que usted quería que hiciese.

Pero fijemos la atención (otra vez) en el tiempo.

¿Que probabilidad cree usted que hay de que un determinado conjunto de entradas referentes al tiempo dé lugar a dos o más juicios idénticos basados en un determinado conjunto de reglas?. Es difícil calcular una respuesta exacta, pero, desde luego, tiene que ser aproximadamente cero.

Los valores de las variables que usted introduce varían de modo continuo. Las variables de tipo REAL pueden tomar un número infinito de valores (sin considerar las limitaciones de la máquina) entre dos valores cualesquiera. Tome varias de estas variables y combínelas con una determinada regla que también está formada por variables de tipo REAL. Tome otra regla diferente, también con variables de tipo REAL, y vea si puede dar el mismo resultado. En general, no. Aunque los dos resultados sean muy parecidos, es altamente improbable que sean idénticos.

Más aún, si vuelve usted a pensar en las reglas como un método para colocar una superficie entre los diferentes resultados, observará que no importa dónde se encuentra exactamente la superficie, siempre y cuando separe los resultados. Del mismo modo, tampoco importa qué valores hay exactamente en las diferentes reglas, siempre y cuando funcionen. Así, esos valores pueden modificarse ligeramente, eliminando las igualdades que pudieran existir, sin que con ello se perturbe el funcionamiento principal del sistema.

Naturalmente, puede decir usted que no se necesita que los valores sean exactamente iguales para que el experto se reserve el juicio, sino sólo parecidos. Pero, ¿cómo de parecidos?, ¿Y cómo se representa en una máquina?. Si usted divide todas sus variables por 100, por ejemplo, antes de aplicar las reglas, las diferencias entre los valores dados por cada regla se reducirán en un factor de 100, y con ello las reglas serán más semejantes, o, al menos, darán resultados más parecidos, y así, seguramente, el experto podría reservarse el juicio con mayor facilidad. Evidentemente, esto no es correcto. Pero ¿cuál sería el modo correcto de proceder?.

Puesto que hasta ahora venimos pensando en términos de categorías, podríamos introducir un resultado adicional y llamarle «No Sé». Pero, ¿cómo entrenaría usted al experto para ese resultado?. Lo más importante en las sesiones de entrenamiento es que usted conoce con certeza los resultados. Si el experto le pregunta si «No Sé» es el re-

sultado correcto, ¿qué le contestaría usted?. ¿Se daría, quizás, por vencido y le diría que usted tampoco lo sabe?.

En esencia, el problema no es decidir si el experto realmente conoce la respuesta o no, sino decidir en qué medida tiene confianza en cada uno de los posibles resultados. Su respuesta final consiste en sugerir aquel resultado en el que más confianza tiene. Pero, en el caso de que esta confianza fuera parecida a la que tiene en algún otro resultado, usted preferiría que el experto se reservara el juicio, o, al menos, que le informase de ello.

Y tal cosa no es posible con el experto que hemos construido. Nuestro sistema experto está diseñado para aceptar cualquier entrada de cualquier problema y dar la mejor respuesta. Si quiere usted saber cuánta confianza puede depositarse en esa respuesta, sólo hay una forma de proceder: darle algunos datos y ver con qué frecuencia acierta.

Con el ejemplo del pájaro/avión/planeador encontramos que, después del entrenamiento, el experto obtiene siempre la respuesta correcta. Así, su opinión se expresa con probabilidad 1. Con el ejemplo de las predicciones meteorológicas, acierta el 75 por ciento de la veces. Así, su respuesta se expresa con probabilidad 0.75. Pero ésas son cifras para las respuestas en general, no para respuestas aisladas, específicas. Puede darse una respuesta con una certeza absoluta y otra con un alto grado de incertidumbre. Y el problema es que no sabemos cuál es cada una. Y tampoco tenemos, en este caso general, ninguna forma de saberlo.

6.3 EL PROBLEMA DE LA DISTANCIA

Seguramente, alguien podría ahora pensar en lo siguiente: Tenemos un conjunto de reglas aceptables en nuestro experto, construidas tras una larga sesión de entrenamiento. Introducimos entonces otro elemento para que el experto decida. Este podría entonces listar todos los resultados posibles, junto con los valores que dan las reglas para esa entrada concreta. El experto elegiría el resultado que tuviese el mayor valor. Este valor, y los valores correspondientes a los resultados no elegidos, ¿reflejarían quizas las probabilidades de que cada resultado sea el correcto?

Tristemente, no. No las reflejaría. El experto ha sido entrenado para elegir el valor mayor, pero no se le ha dicho nada sobre lo que esos valores significan realmente en términos de probabilidades. El mayor valor indica el resultado más probable, pero en qué medida es más probable, es un absoluto misterio para el experto.

Todo ello está relacionado con el problema de la distancia y de las medidas de distancia, y eso es lo que vamos a considerar ahora.

En el sistema de aprendizaje actual, el experto tiene que colocar una valla que delimite entre varios resultados diferentes. No importa en dónde se encuentra exactamente la valla, siempre y cuando separe con precisión los resultados. Lo que importa es, simplemente, a qué lado de la valla caen las variables, y no a qué distancia de ella se encuentran. Y, en este sentido, al experto no se le ocurre pensar en distancias.

Como ayuda para clarificar las ideas, imaginemos que tenemos ante nosotros el mapa de un extraño país, como el que usted puede ver en la Figura 6.1. Tiene dos ejes (Norte-Sur y Este-Oeste) y dos posiciones marcadas en él (X e Y), y tiene unas líneas de nivel para indicar la altura del terreno. Las posiciones X e Y, como vemos por las líneas de nivel, están ambas situadas en las cimas de sendas montañas.

Por tanto, tenemos la Montaña X y la Montaña Y, y, como vemos en el mapa, el punto P se encuentra, más o menos, en la Montaña Y, lo que significa que cualquiera puede mirar al mapa y decir a qué Montaña pertenece el punto P.

Establecido este mapa, podemos fijarnos ahora en los distintos modos en que nuestro sistema experto podría resolver el problema.

Utilizando el algoritmo de aprendizaje, todo lo que hace el programa es colocar una valla delimitadora entre las dos montañas, y, siempre y cuando el punto P esté en el lado de la Montaña Y, la valla estará en una posición correcta, porque el punto P se clasificará correctamente. Pero eso no nos dice en qué medida el punto P pertenece a la Montaña Y: podría encontrarse justo en el pico, en el mismo punto Y o abajo en las estribaciones. Todo lo más que puede decirnos es que el punto Y no forma parte de la Montaña X.

Ello nos conduce a la medida de distancias.

La forma más fácil, y la primera que a uno se le ocurre, es medir las distancias de P a X y de P a Y, y clasificar P según cuál de ellas sea la menor. No es un mal método éste, de modo que vamos a probarlo. Desafortunadamente, no tiene sentido hacer uso de una regla graduada, porque lo que tiene el ordenador no es un mapa como tal, sino una serie de referencias al mapa que indican medidas a lo largo de los dos ejes. Así, el punto X tiene (NS_x,EO_x) como coordenadas Norte-Sur y Este-Oeste, el punto Y tiene (NS_y,EO_y), y el punto P tiene (NS_p,EO_p).

Para calcular, por ejemplo, la distancia de P a X, calculamos:

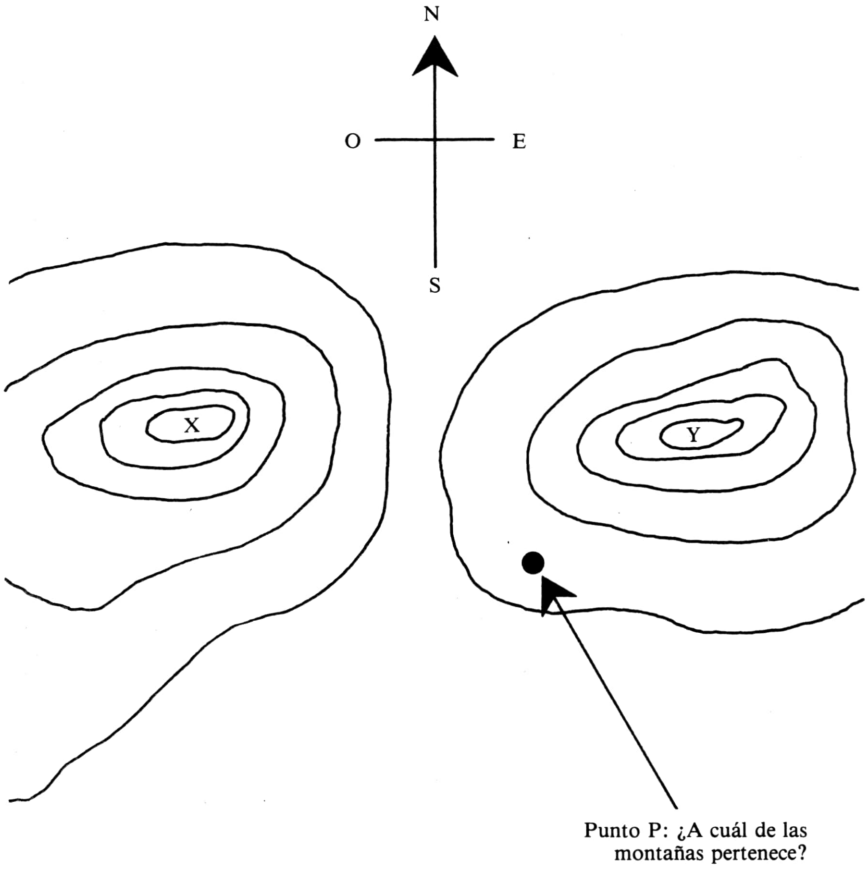


Fig. 6.1
Una analogía física de las medidas de distancias

$d^2 = (NS_x - NS_p)^2 + (EO_x - EO_p)^2$, que es la medida de la Distancia Euclidiana (elevada al cuadrado para que todo resulte positivo) de P a X. Calculamos del mismo modo la distancia de P a Y, y habremos conseguido así un Clasificador de Distancia Euclidiana Mínima para el punto P.

El hecho de que el ejemplo de las montañas se desarrolle en el espacio real y nuestros problemas tengan existencia en un espacio de descrip-

ción de dimensión n, no afecta al modo de proceder. Todo lo que hemos de hacer es calcular la «posición» de cada variable para cada una de las categorías que corresponden a un resultado. Tomando el ejemplo del pájaro/avión con dos variables, pico y motor, tendremos las coordenadas (1,0) para un pájaro (pico y no motor), y (0,1) para un avión (no pico y motor). Dado un objeto P que tenga como coordenadas (1,0), podemos clasificarlo como pájaro, porque ello da la medida mínima de Distancia Euclidiana de entre las dos posibilidades. De hecho, si en la matriz R guardamos las «posiciones» de cada resultado, podemos introducir un nuevo objeto en el vector V y calcular:

```
FOR I=1 TO V
D=D+(V(I)-R(I,J))^2
NEXT
```

y escoger aquel resultado J con el que se obtenga el valor mínimo de D. Ningún problema. Esta forma de proceder podría ser, de hecho, más rápida de programar que el algoritmo de aprendizaje.

La observación que hay que hacer ahora es que, quizás sin percatarnos de ello, hemos sido muy imprecisos en cuanto a lo que verdaderamente le pedimos al experto que haga. Cuando preguntábamos a qué categoría pertenece P, en realidad no habíamos definido las categorías en absoluto, porque lo mismo podíamos estar preguntando: «¿cuál es la distancia más corta, P-X o P-Y?» que: «¿a qué montaña pertenece P, a la X o a la Y?». Y mientras que la distinción puede que no importe realmente mucho en el ejemplo del pájaro/avión, podría ser esencial en el pronóstico meteorológico, por ejemplo.

En el ejemplo del pájaro/avión, todo lo que hacemos es encontrar el mínimo de las distancias de P a X y a Y, y llevar a cabo los cálculos sin mayores problemas. Podemos hacerlo porque las variables son funciones puntuales: sólo existen en un único punto, y la naturaleza del terreno que separa los puntos, podríamos decir, no hace al caso.

Si trabajamos con variables continuas, como las cifras de precipitación, ya no tendremos funciones puntales, sino variables continuas que definen un terreno intermedio. Y, además, se hacen precisos ciertos cálculos adicionales, simplemente para determinar dónde se encuentran realmente las cimas de las montañas.

En general, la mejor forma de calcular las posiciones de las cimas de las montañas es obteniendo los valores medios (o medias) de las variables para cada resultado. Si tenemos n observaciones para un resultado determinado, entonces la media, m, es:

$$m = (x_1 + x_2 + x_3 + \ldots + x_n)/n$$

fórmula bastante conocida.

Así, ya sean discretas o continuas las variables que tenemos, podríamos proceder siempre calculando los valores medios de todos los resultados , poniéndolos en una matriz R y programando un Clasificador de Distancia Euclidiana Mínima.

Sin embargo, aún persiste el problema de que lo que hemos calculado son las distancias a las cimas de las montañas; no hemos contestado exactamente a la pregunta de a qué montaña pertenece P.

Para ver por qué esto podría ser importante, supongamos que las montañas fuesen de diferentes tamaños: una podría ser una montaña propiamente dicha y la otra una pequeña montañita hecha por un topo. Supongamos que ahora el punto P sólo está a un metro del centro de la montañita del topo y a 200 metros del centro de la montaña. ¿Pertenece el punto P a la montaña o a la montañita? Utilizando el Clasificador de Distancia Euclidiana Mínima, no pertenece a la montaña, sino a la montañita. Pero ¿es correcto decir tal cosa? Después de todo, un metro es bastante distancia comparado con las dimensiones de una montañita de topo, mientras que 200 metros al centro de la montaña podrían ser la mitad del camino para subirla. Ya ve usted el problema.

Aquí es donde entran en juego las líneas de nivel. Si estuviésemos fijándonos en un mapa real, tendríamos en cuenta intuitivamente el tamaño de las montañas para responder a la pregunta, y eso es lo que hemos de tratar de hacer ahora con el problema de la clasificación de objetos.

En términos de variables, las líneas de nivel representan la función de distribución de probabilidad (fdp) de la variable. Un ejemplo es la fdp normal que, vista «de lado», presenta el aspecto de la Figura 6.2. El punto X es el valor medio de las variables que tengan esa fdp, y, si midiésemos en dos dimensiones, visto desde la cima parecería el dibujo de una montaña con líneas de contorno.

Ahora bien, asociada con cada fdp, hay una «desviación típica», s, que es la raíz cuadrada de su varianza. La desviación típica es:

$$s = \sqrt{((x_1-m)^2 + (x_2-m)^2 + (x_3-m)^2 + \ldots + (x_n-m)^2)/n}$$

De modo que primero se calcula la media, m, para las n observaciones de cada variable, y luego se calcula la desviación típica, s.

La desviación típica es una medida del «tamaño» de la montaña: mi-

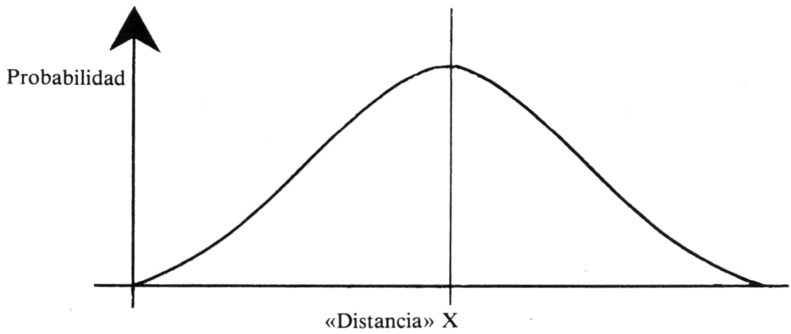


Fig. 6.2
La función de distribución de probabilidad normal

de la dispersión de cada variable. Por tanto, podemos hacer todas las montañas del mismo tamaño, dividiendo todas las medidas por s. Esto es lo que se llama «escalado a varianza unidad».

Partiendo de un conjunto de datos, es posible calcular la media y la desviación típica y obtener una nueva medida de «distancia» así:

$$d^2 = ((x-m)/s)^2$$

lo cual conduce a un clasificador basado en la distancia mínima escalada a varianza unidad.

Esto, ciertamente, servirá de ayuda al experto para que tome decisiones, pero ¿le permitirá al experto decir con qué probabilidad son ciertas sus afirmaciones?.

Bueno, pues la respuesta es: quizás.

Suponga que las variables siguen, realmente, una distribución normal (es decir, que tienen una fdp normal). La medida $d = \sqrt{d^2}$ está tabuda para la fdp normal, y se puede mirar en las tablas (o construir uno sus propias tablas en el ordenador, una tarea no muy fácil) para ver cuál es exactamente la probabilidad de que, con un determinado valor de d, la nueva observación pertenezca a la montaña X (o, en términos estadísticos, para ver la probabilidad de que la observación provenga de una determinada población). Si se hace tal cosa para cada resultado, el sistema experto podrá hacer afirmaciones asignándoles probabilidades.

Sin embargo, el problema que surge es que muchas variables no siguen la distribución normal, y puede ser bastante difícil determinar exactamente cuál es su fdp.

Volviendo al ejemplo del pájaro/avión con dos variables, Pico y Motor, es de todo punto evidente que tales variables no siguen una distribución normal. Como funciones puntuales que son, se parecen a las montañas tanto como los postes de telégrafos: tienen todo concentrado en un sitio. De hecho, ello hace las cosas más fáciles en este caso particular, porque no tenemos que utilizar las tablas: la probabilidad es 0 ó 1, dependiendo de qué variables y qué resultados tenemos, de modo que el sistema puede hacer afirmaciones con probabilidades exactas acerca de sus opiniones. La medida de la «distancia» siempre será 0 o mayor que 0, y la probabilidad de que una clasificación sea correcta será 1 si la distancia es 0, y 0 si la distancia es mayor que 0. Fácil.

El verdadero problema surge cuando, sencillamente, no conocemos la fdp de una determinada variable, y éste podría ser el caso, por ejemplo, de las cifras de precipitación. Podemos estar bastante seguros de que no es una variable con distribución normal, porque no es simétrica en torno a su valor medio. Y sabemos que no es una función puntual, porque es continua. Por tanto, ¿qué es?. Y si conociésemos la respuesta y la introdujésemos en nuestro experto, tendríamos un sistema experto muy bueno para la predicción meteorológica, pero probablemente de poca utilidad para cualquier otra cosa que tenga una fdp diferente (ver Figura 6.3).

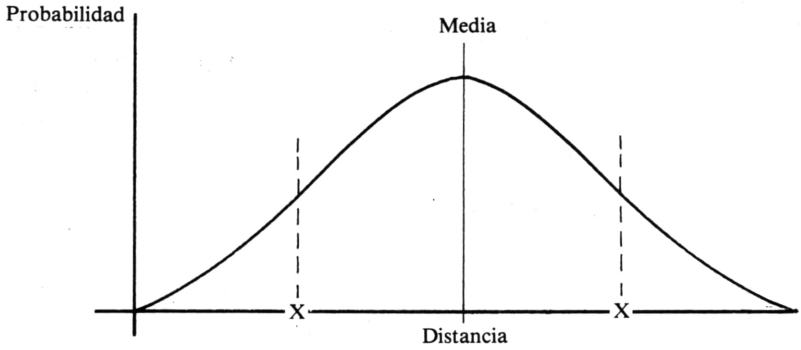
El concepto de medidas de distancias es un buen concepto, pero el traducir esas distancias en medidas de probabilidades puede ser una tarea bastante compleja, porque distancia y probabilidad no son en absoluto la misma cosa.

El problema de la distancia es realmente un problema muy general, y el problema de la similaridad se transforma en él.

Supongamos que tenemos un determinado conjunto de variables y que queremos decidir un resultado. Lo que podríamos hacer es repasar el conjunto de valores con los que el experto ha sido entrenado para ver cuántos de ellos son idénticos a los valores que acabamos de dar. Podríamos entonces hacer que el experto emitiese un juicio escogiendo aquel resultado que contenga el mayor número de ejemplos que tienen exactamente los mismos valores de las variables. Y éste no es un mal método, si los datos se adaptan a ello.

Tiene la ventaja, en su forma más sencilla, de no depender de la medida de distancias: podríamos utilizar categorías de elementos, o cual-

a) En una fdp bien formada, como la fdp normal, las probabilidades varían de una manera muy regular en función de su distancia a la media.



b) Cuando la fdp no está tan bien formada, la relación puede ser mucho más difícil de definir. En este ejemplo «sesgado a la izquierda», los valores que son inferiores a la media pueden ser mucho más probables que algunos de los valores superiores a la media.

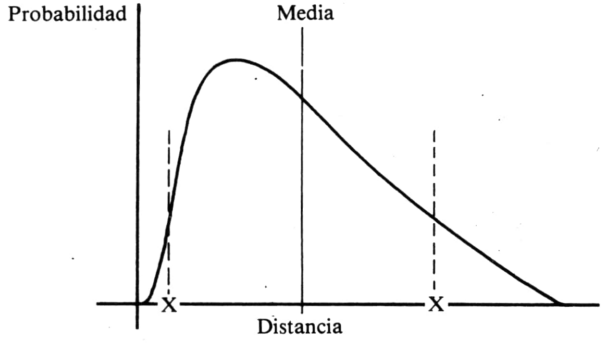


Fig. 6.3
Funciones de distribución de probabilidad simétrica y asimétrica

quier otro método que prefiramos para describir los datos. Para el ejemplo del pájaro/avión/planeador funcionaría muy bien.

Pero volvamos al tiempo. En este caso, la posibilidad de encontrar aunque sólo sea un conjunto de valores idénticos es muy remota. Por

tanto, ¿cómo mediremos la similaridad?. Ello nos devuelve a la medida de distancias y todos sus problemas. Es ciertamente una pena, porque, si el método funcionase, podríamos asignar realmente probabilidades a los distintos resultados basadas en el número de casos similares incluidos en cada categoría. Ciertamente, si tiene usted un problema específico que se adapta a este método de trabajo, encontrará que es de uso sencillo, fácil de comprender y razonablemente preciso. El tipo de situación en la que podría funcionar el método, sería aquella en la que las variables consisten en un cierto número de categorías, pero que esas categorías no definen con precisión un resultado particular. En el ejemplo del pájaro/avión/ planeador, las categorías definen exactamente el resultado. Sin embargo, si tiene usted una lista de síntomas médicos que indican alguna enfermedad pero que no apuntan con absoluta certeza a una enfermedad exacta, entonces este método daría también buenos resultados. Podría, de hecho, dar tan buenos resultados como nuestro método original (lo cual es una buena razón para defender aquel método), pero también sería capaz de indicar en qué medida piensa que sus juicios son correctos.

Obviamente, una variante de esta técnica consiste en utilizar el método original para establecer un juicio y luego explorar los registros anteriores para comprobar si hay ejemplos concretos que fuesen idénticos, de modo que pudiera hacerse una indicación sobre otros juicios posibles.

6.4 LA COMPRENSIÓN DEL PROBLEMA

Lo esencial del enfoque que hemos seguido para construir un sistema experto ha sido elaborar un método que funciona con un grado razonable de fiabilidad en todas circunstancias. El método debe generar un resultado medianamente aceptable, independientemente del problema que usted le plantee, y, lo que es más importante, independientemente de si usted comprende realmente o no el problema.

Tomemos el pronóstico del tiempo. Puede ser que usted trabaje en el Centro Meteorológico, pero probablemente no es así, y probablemente no sabe cómo predecir el tiempo. Pero, aún así, usted puede desarrollar un sistema experto que le ayude a hacerlo.

Lo mismo se aplica a otros muchos problemas en los que podría usted pensar. El mismo sistema experto básico puede utilizarse para atacar una gran variedad de problemas.

Pero si hay un problema que usted comprende realmente bien, entonces tiene la posibilidad de obtener resultados mucho mejores y preci-

sos si construye un sistema a la medida para él. Lo que puede señalarse es que, si verdaderamente comprende usted su problema, entonces no necesita de nadie que le diga cómo proceder. ¿No es cierto?

7. UN EXPERTO EN TODAS LAS COSAS DEL MUNDO ENTERO CONOCIDO

7.1 NODOS

Una vez que ha llegado hasta este punto en la construcción de su sistema experto, usted puede considerar una aplicación muy especial para él: hacerlo experto en Todas las Cosas del Mundo Entero Conocido. Quizás le parezca difícil, pero en realidad no lo es. Sólo se trata de dedicarle tiempo.

Empiece, como antes, inicializando el sistema, ahora con siete variables y cinco resultados:

Variables	*Resultados*
Alas	Pájaro
Pico	Avión
Motor	Planeador
Temp. Mín.	Mañana lloverá
Temp. Máx.	Mañana no lloverá
Precipitación	
Horas de Sol	

Ahora comience una sesión de entrenamiento. Enséñele primero los secretos de los pájaros, los aviones y los planeadores, proporcionándole ejemplos de cosas que tengan o no tengan alas, picos y motores. Si el experto le pregunta sobre las otras variables, respóndale con un cero o con algo que no implique un compromiso. Cuando el sistema parezca haberlo entendido, instrúyale sobre el tiempo, y si pregunta sobre alas, pico o motores, responda de nuevo con ceros. Déjele funcionar y ensayar durante algún tiempo.

Ahora, lo crea usted o no, tiene un sistema que es experto en la predicción del tiempo y en la identificación de objetos volantes simultáneamente.

La generalización de la situación para cubrir todo el espectro del Conocimiento Humano es obvia. No tiene más que seguir añadiendo nuevas variables, nuevos resultados y nuevos ejemplos de entrenamiento. Y, finalmente, el experto se hará experto en todo. Dentro, por supuesto, de las limitaciones de la capacidad de la memoria de su ordenador.

«¡Pero esto es verdaderamente maravilloso!», exclamará usted.

Bueno, sí, claro que lo es. Mejor no perder el tiempo diseñando sistemas de poca monta, cuando pueden resolverse todos los problemas del mundo. Pero posiblemente encuentre usted que el sistema resulta un poco lento. Y posiblemente se irrite cuando quiera tener un pronóstico del tiempo, y él siga preguntándole acerca de picos y motores. Porque una de las dificultades con este esquema es que, tal como está, el experto no sólo es capaz de resolver cualquier problema del Mundo Entero Conocido. Cada vez que conecte usted el funesto aparato, va a tratar de resolver todos los problemas. Su intención es buena, por supuesto, pero la admiración que uno siente por él pronto se desvanece.

La solución, evidentemente, consiste en tener dos versiones del experto. Una adecuada a los objetos volantes y otra a las predicciones del tiempo. Y otras versiones adecuadas para otras cosas, si quiere. Y no las conecte entre sí, porque no tiene sentido hacerlo.

Lo cual nos lleva a los nodos.

Porque lo que hemos sugerido es la idea de nodo. Hay un nodo para la identificación de objetos volantes y otro para el pronóstico del tiempo. Los dos nodos no están conectados, porque no tienen necesidad de ello. Pero cada nodo tiene varias entradas (variables) y varias salidas (resultados), y podrían haberse conectado, si hubiésemos querido hacerlo.

Por ejemplo, supongamos que tenemos dos nodos expertos en pronóstico del tiempo. Podríamos haber conectado la salida de un nodo a una de las entradas del otro. Y entonces podríamos utilizar el primer nodo para predecir, por ejemplo, el tiempo de esta tarde, y el segundo para predecir el tiempo de mañana.

Entonces, cuando el experto hiciese una predicción para esta tarde, podría usarla como una entrada para su predicción «a largo plazo» (¡mañana!).

Pero volvamos a ese gran experto en absolutamente todo. Tal nodo no es más que el ejemplo extremo de un proceso paralelo. Usted le dice todo lo que él quiere saber, y él, a su vez, le dice algo a usted. Al dividirlo en dos nodos más pequeños (predicción del tiempo e identificación de objetos volantes), lo convertimos en dos procesos paralelos. Como no se ejecutan simultáneamente, son dos procesos paralelos organizados secuencialmente. Y como no tienen nada que ver entre sí, no se conectan. Pero si se conectasen, formarían una especie de proceso secuencial compuesto por una serie de nodos (dos en este caso) con una estructura de árbol, si prefiere usted verlo así.

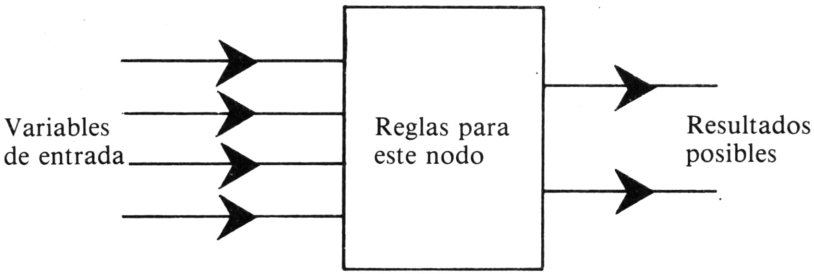


Fig. 7.1
Un nodo único

Cada nodo podría inicializarse y entrenarse individualmente, de modo que cada uno funcionaría tan bien como si estuviese solo.

Y aunque, en teoría, esa disposición no tiene por qué ser mejor que un gran proceso paralelo, en la práctica puede ahorrarle algún tiempo, así como proporcionarle alguna información adicional acerca del funcionamiento del sistema.

Pensemos de nuevo en el tiempo, por cambiar de ejemplo.

Actualmente, dispone usted de un nodo que le dice si mañana va a llover o no. Muy bien. Pero también podría decirle otras cosas. Como, por ejemplo, si hoy hace calor o no. Ahora bien, quizás alguien piense que cualquier idiota es capaz de responder a ello. Cierto, pero estamos con un ordenador, por lo que las cosas son un poco más complicadas.

Inicialice otro nodo y empiece a entrenarlo. Haga que diga que hoy hace Calor, si la temperatura es superior, por ejemplo, a 10.°C. El siste-

ma conseguirá entenderlo sin grandes dificultades, aunque usted, personalmente, tenga algunas dudas sobre si 10.°C representan calor o no.

Vuelva ahora a su nodo anterior y déle cinco variables: temperatura máxima, temperatura mínima, precipitación, horas de sol y calor. «Calor» (hoy) es la salida del primer nodo y, evidentemente, es 1 si la temperatura máxima está por encima de 10.°C, y 0 en caso contrario. Entrene al segundo nodo para predecir si mañana lloverá o no. El proceso es el indicado en la figura 7.2.

Observará usted que, por alguna extraña razón, el segundo nodo se comporta ahora de modo diferente a como lo hacía antes, cuando sólo tenía cuatro variables.

Es como si tuviese alguna información adicional. Pero todo lo que usted le ha dado es una variante de la temperatura exacta, que era algo que ya tenía.

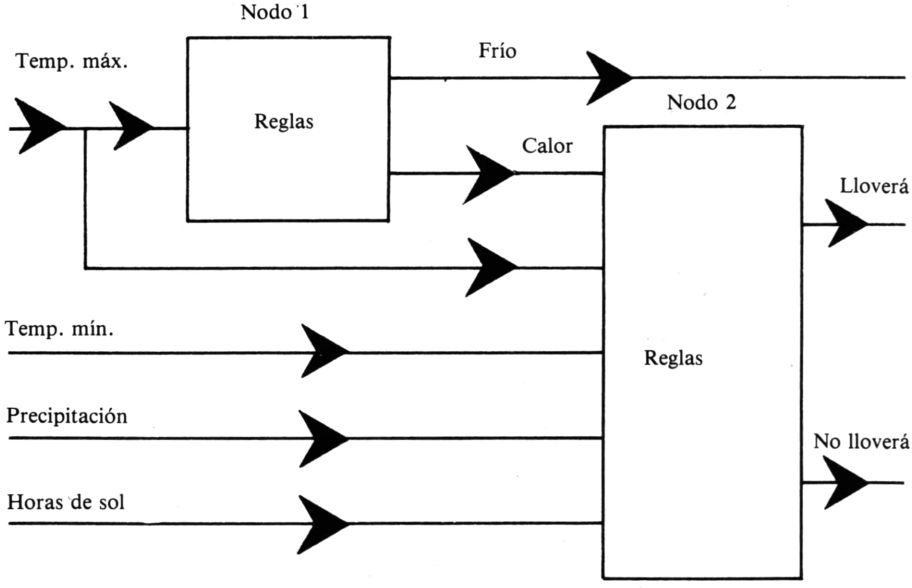


Fig. 7.2
Cómo añadir un nodo adicional

Se pueden combinar varios nodos para formar una red que proporcione conclusiones intermedias y entradas para otros nodos

Pues no es así exactamente. Usted le ha proporcionado cierta información adicional. Ese elemento sobre el calor representa algo más de lo que ya tenía antes. No mucho más, pero sí un poco. Hay que observar que el Calor, tal como lo hemos definido, es una transformación no lineal de la temperatura máxima. Y como el experto sólo trabaja con transformaciones lineales, se le ha dado algo que no tenía antes y que jamás habría podido deducir por sí mismo.

Si no está usted muy convencido, una manera de pensar en ello es la siguiente: suponga que el experto sólo tiene las cuatro variables: temperatura mínima, temperatura máxima, precipitación y horas de sol. A partir de ellas, elabora un conjunto de reglas para predecir el tiempo. Le pide usted entonces un pronóstico, e, ignorando de momento las otras tres variables, toma la temperatura máxima y la multiplica por un número que tiene en su conjunto de reglas. De acuerdo con el resultado, hace su predicción.

Añada ahora el Calor. Su valor es 0 si la temperatura máxima es inferior a 10.°C, de modo que, por debajo de esa temperatura, todo se calcula como antes.

Pero en cuanto la temperatura supera los 10.°C, la variable Calor toma el valor 1, que se multiplica por algún número del nuevo conjunto de reglas y se añade a los resultados finales. Así, cuando la temperatura máxima supera los 10.°C, de pronto, por así decirlo, se altera bruscamente el proceso de decisión. Matemáticamente, esto se llama una función escalón, función que es no lineal.

Por tanto, ahora tiene usted una variable adicional, una variable realmente nueva, y, obviamente, podría usted considerar que tiene cinco variables, y siempre que el experto le preguntase si hace calor hoy, no tendría más que mirar la temperatura máxima, y, de acuerdo con ella, darle un 0 o un 1.

Pero también puede usted conseguir que el experto lo haga en su lugar, poniendo un nodo especial para ello. Este nodo no necesita realmente más que una variable de entrada (la temperatura máxima), y se puede entrenar para que reconozca el Calor y obtenga la respuesta a esta cuestión, que se utilizará como entrada al siguiente nodo, el que predice la lluvia. Es algo así como manipular los cables de una cadena de alta fidelidad. Se coge el cable de salida de una unidad y se enchufa en otra unidad. Y la temperatura máxima es un elemento con cables que van a dos unidades. Y, como en una cadena de sonido, se puede seguir añadiendo nuevos cables y enchufando y desenchufando hasta cansarse. Afortunadamente, la cosa funcionará antes de que el esfuerzo acabe con usted.

Conseguir esto implica una modificación del programa, pero no demasiado grande. Suponga que decide usted que su programa va a tener N nodos. Entonces, al especificar sus variables y reglas, puede escribir: DIM V$(V,N), Q$(Q,N), R(V,Q,N), V(V,N).

Dicho de otro modo, agregue una dimensión a cada variable (la mayoría de las cuales ya eran vectores o matrices), con lo que, dando un valor de N, podrá referenciar N nodos diferentes.

Supongamos que se llega a un nodo que establece un juicio Q$(J,N1) (resultado J en el nodo N1). Puede tenerse luego una pequeña rutina que explore todas las otras variables para ver si este resultado es una variable de entrada para cualquier otro nodo. Si lo es, entonces, suponiendo que V$(I,N2) = Q$(J,N1), haremos V(I,N2) = 1 para la variable I del nodo N2.

Viene a ser algo así como una exploración automática para ver si en el sistema hay interruptores que pueden cerrarse.

Y el hecho de cerrar un interruptor viene a ser equivalente a proporcionarle al experto otro valor de variable, de manera que entonces él por su cuenta es capaz de comprobar si con ello puede extraer nuevas conclusiones. Si puede, tendrá otro resultado que le servirá como posible entrada para otros nodos. Y así sucesivamente, hasta que llega a algo que usted juzga útil. Como la respuesta al problema original que le planteó.

Desde luego, la verdadera diferencia en el comportamiento del experto que tiene más de un nodo radica en el hecho de que en cada nodo se realizan, normalmente, transformaciones no lineales; no es la simple presencia de nodos lo que le hace actuar de un modo diferente.

Pero hay una ventaja adicional. Los nodos le proporcionan a uno una idea de lo que está pasando por la cabeza del experto. Porque cada uno de esos nodos da un resultado que se puede considerar como una solución intermedia. Por ejemplo:

Entrada:	Valor de la Temperatura máxima hoy
Salida:	¡Hace calor hoy!
Entrada:	Temperatura mínima, Precipitación, Horas de sol
Salida:	Apuesto a que mañana llueve (empiezo a sentirme melancólico)

Las respuestas exactas del experto dependen de cómo lo programe usted, pero así es más o menos como podría responder, y, desde este punto de vista, el asunto de si hoy hace calor o no ya no parecerá tan tri-

vial. Puede llegar a conseguirse que el experto hable casi como lo haría una persona. (Siempre y cuando, desde luego, las personas tuviesen la costumbre de hablar escribiendo sus palabras en una pantalla cada vez que se les mencionara el tiempo).

Indudablemente, cualquier aplicación que a usted le interese tendrá muchos más nodos y muchos más resultados, y podrá tratar de algo mucho menos trivial, como, por ejemplo, Neurocirugía. Si es éste el caso, puede evitarse muchos problemas haciendo que el experto le comunique algunas conclusiones intermedias antes de que al final le aconseje: «Córtele la cabeza al paciente».

7.2 LAS VARIABLES, TAL COMO ESTÁN

Por ahora, y a menos que haya estado usted muy concentrado y tomando notas al mismo tiempo, probablemente habrá obtenido una asombrosa colección de afirmaciones poco relacionadas entre sí, y amontonadas bajo el encabezamiento de Sistemas Expertos. Es verdaderamente sorprendente cómo puede enredarse uno con esas cosas. Todo lo que usted ha hecho ha sido escribir un trozo de programa, añadirle otro trozo, modificar el primero, pensar en alguna otra cosa, irse a tomar café, tener una brillante idea, añadirla,...

Lo que realmente necesita usted es un poco de orden. Helo aquí: una colección de las variables más importantes que puede haber utilizado hasta este momento.

```
DIM V$(MV,N)
```

Esta es la lista de todos los nombres de las variables que tiene usted en su sistema experto. Para cada nodo hay una lista. El número máximo de variables que puede tener en cualquier nodo es MV, y el número de nodos es N.

Cuando usted empieza a inicializar el sistema y quiere DIMensionar los vectores y matrices, lo primero que tiene que hacer el sistema es preguntar:

```
INPUT «¿CUANTOS NODOS HAY?»; N
```

y después:

```
INPUT «¿CUAL ES EL NUMERO MAXIMO DE VARIABLES
QUE PUEDEN TENERSE EN CUALQUIER NODO?»; MV
```

o frases parecidas.

Evidentemente, al proceder de este modo se va a gastar mucho espacio en memoria para guardar las matrices, a menos que ocurra que todos los nodos tengan el mismo número de variables de entrada. Pero, si bien esto puede evitarse, proceder de otro modo nos complicaría la vida. Sería más económico disponer de un vector para cada nodo, pero es difícil ver cómo podrían llegar a DIMensionarse todos ellos en BASIC sin saber, en el momento de escribir el programa, cuántos nodos va a haber.

Por tanto, en cada nodo podrá usted tener hasta MV variables.

DIM V(MV,N)

Esta es la lista de valores de las variables en cada uno de los N nodos, en correspondencia con los nombres de las variables.

DIM VC(MV,N)

Esta es una especie de matriz «señal» que indica cuáles son las variables cuyos valores ya se han introducido en el proceso de decisión y cuáles no. Su valor es 0 para elementos que ya se han introducido y 1 para los que todavía esperan. La secuencia de acontecimientos consiste en que usted introduce los nombres de las variables en V$(x,y), el sistema pide el valor de alguna V$(x,y) para la cual VC(x,y) = 1, introduce este valor en V(x,y) y luego hace VC(x,y) = 0.

DIM VR(MV,N)

Esta es la matriz que mantiene una «puntuación» de cada una de las variables en cada uno de los N nodos. El sistema la utiliza para determinar cuál es la siguiente variable cuyo valor va a pedir. Normalmente preguntará por aquella V$(x,y) que tenga el mayor valor en VR(x,y).

DIM M(2,MV,N)

Esta matriz guarda los valores mínimos y máximos de las variables V$(x,y). Así, el mínimo valor que puede tener V(x,y) viene dado por M(1,x,y), y el máximo por M(2,x,y). Es usted quien debe proporcionar esos valores, normalmente cuando introduce los nombres de las variables en V$(x,y). Tales valores se utilizan para ayudarle al experto a evaluar la importancia relativa de cada variable, a fin de llegar a tomar una decisión, y se necesitan para calcular VR(x,y). Si no sabe usted los valores máximos y mínimos que las variables pueden tomar, trate de imaginárselos. Al fin y al cabo, algunos límites tienen que tener.

DIM VA(N)

Este vector contiene el número de variables que hay en cada uno de los N nodos. El sistema ya conoce el número máximo (MV), pero tiene que saber cuántas hay realmente, y aquí es donde están esos números. En una situación (como la que teníamos anteriormente) en la que sólo hay un nodo, esta información está contenida en una única variable, V. Por tanto, para varios nodos, dejamos V y utilizamos VA(x). No podemos utilizar V(x), porque ya tenemos V(x,y), y no se puede utilizar el mismo nombre para dos variables dimensionadas. Si usted ya tuviera escrito un experto de un solo nodo y decidiera convertirlo en un experto con varios nodos, podría ahorrarse un poco de trabajo siguiendo con la variable V, pero antes de hacer uso de ella, tendría que hacer V = VA(x), suponiendo que está en un segmento de programa en el que se trabaja con el nodo x.

DIM Q(N)

Este vector contiene el número de resultados que pueden darse en cada uno de los nodos. Los comentarios son similares a los de VA(N). También puede seguir utilizando la antigua variable Q para el número de resultados, pero haciendo Q = Q(x) cuando trabaje con el nodo x.

DIM Q$(MQ,N)

Esta es la lista de todos los nombres de los resultados que utiliza el sistema experto, del mismo modo que V$(MV,N) era la lista de todos los nombres de las variables.

Lo mismo que con los nombres de variables, MQ es el número máximo de resultados que puede haber en cada nodo, y cuando se inicializa el sistema se puede proceder así:

INPUT «¿CUAL ES EL NUMERO MAXIMO DE RESULTADOS EN CUALQUIERA DE LOS NODOS?»; MQ

la mayoría de los comentarios referentes a V$(MV,N) son también aplicables a Q$(MQ,N).

DIM R(MV,MQ,N)

Este es, realmente, el corazón del sistema: la matriz de reglas que utiliza el experto para establecer juicios. Hay N conjuntos de reglas, uno para cada nodo. Cada conjunto de reglas tiene VA(x) variables que entran en él y Q(y) variables que salen de él. R(MV,MQ,N) se construye entrenando cada uno de los nodos, ya sea a partir de ejemplos introduci-

dos por el teclado o dándole al sistema una lista completa de ejemplos y dejándole trabajar sobre ellos.

DIM D(MQ)

Este vector guarda los valores establecidos por las reglas para cada resultado en uno cualquiera de los nodos. Podría hacerse DIM D(MQ,N) para tener en cuenta los N nodos, de manera que siempre se pudiese disponer de todos los valores de D. Que sea o no necesario hacerlo así depende de cómo se organice la búsqueda a través de los nodos. Si se hace que el experto sólo resuelva un nodo a la vez, entonces no se necesita la dimensión adicional, y, en cualquier caso, siempre se puede volver a calcular D(y) cada vez que el sistema entra en un nuevo nodo.

DIM PD(MQ)

En este vector se guardan los valores posibles de cada regla. Suponga que, en un determinado nodo, sólo se han introducido los valores de algunas de las variables. Entonces, puede calcularse D(y), teniendo en cuenta esas variables, a partir de V(x,nodo) y de R(x,y,nodo). Para estas variables, D(y) y PD(y) son idénticos. Pero, teniendo en cuenta las variables desconocidas, el sistema puede «aventurar» PD(y) a partir de los valores mínimos y máximos que se han puesto en M(1,x,nodo) y M(2,x,nodo). La idea es que el sistema haga una «conjetura óptima» en base al máximo valor que en ese momento tenga en D(y). Entonces, calculará PD(y) suponiendo que el resto de las variables tomarán sus valores mínimos o máximos, en un intento para desacreditar a la conjetura óptima actual. Si de este modo el sistema no puede desacreditar a la elección actual, entonces puede tomarse esta elección como resultado para el presente nodo. Si tal elección puede desacreditarse al encontrar que a algún otro resultado corresponde un valor mayor en PD(y), entonces, evidentemente, se necesitan valores de otras variables.

DIM E(MV + 1,50,N)

En esta matriz se introducirán todos los ejemplos que se quiera dar al experto para que practique con ellos. Tiene espacio para conjuntos de hasta 50 ejemplos en cada uno de los nodos.

Desde luego, podría ser que quisiera usted un número distinto de 50, y también que pensara en otra forma de guardar esos ejemplos. Si tiene usted un disco, se le habrá ocurrido enseguida la idea de guardarlos en ficheros. Si no lo tiene, y además está escaso de espacio en memoria, puede hacer DIM E(MV + 1,x), donde x es el número de ejemplos que quiere usted guardar, y entonces tendrá que cargar sucesivamente la matriz para cada uno de los nodos a partir de una cassette.

En cualquier caso, el sistema necesitará saber cuántos ejemplos hay en cada lote, número que puede usted guardar en E(MV + 1,50,N + 1), actualizando cada vez que añada más ejemplos. Si está usted en una sesión de entrenamiento desde el teclado, puede ir colocando en esta matriz todos los ejemplos que le va dando al sistema, de modo que éste pueda luego trabajar con ellos en la intimidad de sus circuitos. Esto le evitará desgastarse los dedos introduciendo dos veces los mismos ejemplos.

Suponga que ha construido usted un experto de un solo nodo y quiere convertirlo en un experto multinodo. Comience por idear un ejemplo y hacerlo funcionar en el sistema de un solo nodo, lo que le servirá como referencia para el comportamiento posterior del sistema.

Después, recorra la lista de variables y transforme el programa para que tenga N nodos. Inserte instrucciones adicionales que pregunten por N, MV y MQ. Ponga un gran bucle sobre el cuerpo principal del programa que permita pasar por todos los nodos:

```
FOR NO = 1 TO N

cuerpo principal del programa

NEXT NO
```

Luego, haga funcionar el programa con N = 1 y con el mismo ejemplo anterior. Los resultados tendrían que ser exactamente los mismos que antes. No debería advertir ningún cambio. Pero los resultados no van a ser los mismos, porque el programa funcionará mal, avisándole así de que habrá cometido usted algún error. Por tanto, corrija los errores. Probablemente, habrá olvidado transformar alguna V o Q en VA(x) o Q(y).

Si, por algún capricho de la suerte, consigue usted que el programa funcione con un solo nodo, entonces hágalo funcionar con dos, colocando ahora el ejemplo anterior en el segundo nodo. Como antes, nada debería alterarse. Quedará usted asombrado y estupefacto al observar cuántas cosas cambian en estas pequeñas ocasiones en las que nada debería haber cambiado. Es verdaderamente instructivo.

Ejecute de nuevo el programa con dos nodos. Pero ahora ponga su ejemplo en ambos nodos. Lo que usted debería observar es que nada cambia. Por tanto, corrija el programa hasta que sea así.

Llega ahora el gran momento en usted va a unir los dos nodos. Pruebe:

Una vez encontrado un resultado para el nodo NO:

```
FOR I=1 TO N
FOR J=1 TO VA(I)
IF Q$(resultado,NO)=V$(J,I) THEN: V(J,I)=1: VC(J,I)=0
FOR K=1 TO Q(NO)
IF V$(J,I)=Q$(K,NO) AND K <> resultado THEN: VC(J,I)=0
NEXT
NEXT: NEXT
```

Lo que hace este trozo de programa es tomar el resultado actual del nodo NO y explorar toda la lista de nombres de variables para ver si ese resultado coincide con alguna variable de entrada de algún otro nodo. Cuando encuentra una coincidencia, pone a 1 el valor de la variable, y su señal correspondiente, VC(J,I), a 0. El valor 1 es totalmente arbitrario, y corresponde a estados Si/No. Si usted quiere, puede utilizar cualquier otro valor; por ejemplo, el valor máximo, M(2,J,I).

Asimismo, se comparan las variables con todos los resultados del nodo NO que no son el escogido. La idea es que, como todos los resultados son mutuamente exclusivos, si un resultado es cierto, todos los otros deben ser falsos, y entonces se pone a cero VC(J,I) en las correspondientes variables para evitar que el experto pida valores para ellas.

Del mismo modo, cuando proporciona usted un valor de variable, V(VV,NO), puede hacer lo siguiente:

```
FOR I=1 TO N
FOR J=1 TO VA(I)
IF V$(VV,NO)=V$(J,I) THEN: V(J,I)=V(VV,NO):
VC(J,I)=VC(VV,NO)
NEXT: NEXT
```

De esta forma, si alguna variable es entrada de más de un nodo, su valor sólo se pedirá una vez.

La idea global es sacar el mayor partido posible a la información que está usted dando. Si usted da un valor de una variable, el experto, que tiene que trabajar para ganarse la vida, proporcionará entradas a tantos nodos como pueda. Si se ha seleccionado un resultado, éste debe introducirse inmediatamente en todos los nodos que puedan hacer uso de esa información, con el objetivo de deducir otros resultados con la menor intervención posible por parte de usted.

Si ahora inicializa usted de nuevo el sistema con dos nodos y hace que ambos traten idénticamente el ejemplo anterior, observará que,

una vez resuelto el primer nodo, el sistema da inmediatamente la respuesta al segundo. Es decir, da la misma respuesta sin necesidad de preguntarle nada a usted; a fin de cuentas, cuando ha resuelto el primer nodo, el experto tiene ya toda la información para resolver el segundo.

Con ello, usted dispone de un sistema multinodo básico que es suficiente para resolver muchos problemas.

Pruebe a inicializarlo para un problema. Por ejemplo, con dos nodos:

Nodo1

Variables	*Resultados*
Alas	Animal
Metal	Máquina
Pico	

Nodo 2

Variables	*Resultados*
Animal	Pájaro
Máquina	Avión
Pico	Planeador
Motor	

Los detalles precisos del ejemplo no importan demasiado. Lo importante a señalar es que el sistema decidirá primero si se trata de un animal o de una máquina. Luego, haría uso de esa información para decidir si es un pájaro, un avión o un planeador. No preguntaría dos veces por el Pico (esa información se la da usted en el nodo 1). Y no preguntaría en absoluto por Animal o Máquina (lo deduciría por sí mismo en el nodo 1).

7.3 CÓMO RECORRER LOS NODOS

Suponga usted que decide construir un sistema experto que tenga más de un nodo. Tal experto tendría, en tal caso, varios puntos en los que sería capaz de emitir algún tipo de juicio, y, dependiendo de ese juicio, podría entonces seguir estableciendo otros juicios.

El problema que se le planteará cuando construya usted un sistema así, es que el programa tiene que pasar de un nodo a otro. Tendrá que proporcionarle un método para que decida cuál es el siguiente nodo, y el método que adopte influirá en que el funcionamiento del sistema sea más o menos bueno o malo.

Para poner las cosas aún peor, no existe ningún método ideal que valga para todas las situaciones. Dependiendo de lo que usted quiera hacer con el experto, un método podría ser mejor que otro.

El problema es exactamente el mismo de la búsqueda en árboles. O el de la búsqueda en grafos. Y entrar verdaderamente en los detalles de este asunto requeriría la mayor parte de un libro dedicado a él.

Afortunadamente, la forma en que hemos trabajado hasta aquí nos ha legado un tipo de problema bastante impreciso, por lo que también podemos atacarlo de una manera imprecisa.

Para empezar, usted podría recorrer los distintos nodos exactamente del mismo modo que seleccionaba las variables en el caso de un solo nodo.

Recuerde que teníamos un vector, VR(V), que contenía un valor para cada una de las variables. Este valor se calculaba para que fuera la cantidad máxima de variación que cada variable podía introducir en el problema. Se toman los valores mínimo y máximo de una variable, se calcula la diferencia entre ellos y entonces se va a la matriz de reglas y se consideran los valores que hay en ella para esa variable y para cada resultado posible. Se multiplican ambos números, sumando para todos los resultados, y ello da una especie de medida de la importancia posible de tal variable. Se toma la variable más importante y se hace que el experto pregunte primero el valor de esa variable. Si ello no le permite establecer un juicio, se busca la siguiente variable en importancia. Y así sucesivamente.

No hay razón por la que no podamos utilizar el mismo enfoque para un gran número de nodos. Habría alguna diferencia en la programación, porque algunas variables podrían darse en más de un nodo, pero, en general, eso haría que al experto le pareciesen más importantes, y preguntaría primero por ellas. Lo cual parece un enfoque razonable. Consiste, simplemente, en seleccionar para su estudio aquellas variables que proporcionan la máxima información en el mayor número de nodos posible.

Lo que realmente estamos haciendo si procedemos así es olvidarnos un poco de las técnicas ordinarias de búsqueda en árboles. Normalmente, se especifica un método de búsqueda en un árbol relacionado con la estructura de ese árbol. Ya sabe usted cómo: se explora a lo largo de un camino hasta que se llega al final, luego se explora el siguiente camino, y así sucesivamente.

En este caso no estamos en absoluto explorando de este modo. Tenemos etiquetas en cada camino, etiquetas que corresponden a los

nombres que hemos dado a las variables. Lo que decimos es: busquemos la variable más importante y exploremos entonces todos los caminos que utilizan esa variable. Y hay buenas razones para proceder así. Porque, en general, no sabemos cuál es la forma exacta del problema que podría planteársele al experto, por lo que tampoco sabemos cuál es el mejor método de exploración del problema. Sabemos, sin embargo, que en el problema tiene que haber variables, de modo que hacemos caso omiso de la estructura y nos concentramos en las variables propiamente dichas. Este enfoque se suele llamar «encadenamiento hacia adelante».

Es un método que a la larga producirá respuestas, aún cuando implique que haya que dar un valor para todas y cada una de las variables antes de obtener la respuesta específica que se quiere. Sólo que algunos podrían sentirse incómodos con un método que parece proceder sin considerar en absoluto la red con la que está trabajando o los eventuales objetivos que se persiguen. No es tanto un método como una especie de procedimiento impreciso que se distribuye por la red de nodos en base a: «Este trozo parece interesante; veamos el siguiente».

Pero observe que, como el sistema desconoce la forma precisa en que están organizados los nodos, también usted puede hacer caso omiso de la forma en que están organizados. Todo lo que tiene que hacer es especificar sus nodos y dar nombres a las variables y a los resultados para cada uno, y el sistema, al final, conseguirá deducir algo. Sin embargo, esto tiene algunos inconvenientes prácticos.

Un inconveniente de lo más práctico radica en el hecho de que todos los encadenamientos entre nodos se realizan a través de nombres específicos de variables y resultados. Esto significa que, si olvida usted cómo llamó exactamente a alguna variable la última vez, el correspondiente encadenamiento no se realizará. Por supuesto, esto viene a ser algo así como decir que, si se equivoca usted al pulsar las teclas de su ordenador, no va a obtener un resultado correcto. Pero puede ser un verdadero problema, si maneja usted un gran número de variables y además ha tenido un día agotador. En principio, no se puede hacer gran cosa para solucionar el inconveniente. Sin embargo, una buena idea, en la práctica, es hacer que el sistema le permita dar los nombres de las variables mostrándole una lista (menú) y preguntándole qué variable quiere, en lugar de teclearla de nuevo cada vez, con el riesgo de equivocarse.

Un hecho más característico de este método de trabajo es que el experto puede considerar que cierta variable es importante, y, en consecuencia, pedirle un valor para ella. Entonces, usted se da cuenta de que no quiere que le pregunte sobre esa variable. Ello podría ser porque en ese momento usted no tiene un valor para ella, o porque usted sabe que,

aunque el experto no lo sepa, esa información no es útil para el problema particular que se está tratando, y que hubiera sido mejor que le preguntase otra cosa.

Este problema puede superarse muy fácilmente con varios nodos. Suponga que tiene usted N nodos a los que puede acceder colocándolos dentro de un bucle. Así, el sistema considera primero el nodo 1, luego el 2, y así hasta el nodo N.

En cada nodo, el sistema puede utilizar el método anterior para decidir sobre qué variables preguntar y, al encontrar un resultado, comprobar todos los otros nodos para ver si ese resultado es una variable de entrada en alguno de ellos. Ahora bien, el orden exacto para pasar de un nodo al siguiente puede especificarse rígidamente en el momento en que se inicializa el sistema. De este modo, si conoce usted el problema sobre el que quiere entrenar al experto, puede inicializarlo para que trate primero con aquellas cosas que usted personalmente sabe que han de tratarse primero. Y si no sabe usted el orden perfecto, o es indiferente, nada se pierde realmente con ello.

En términos simplificados, el problema consiste en dónde hacer que empiece el experto a trabajar con la tarea que se le ha encomendado, y lo peor que puede pasar es que usted tampoco sepa por dónde hay que empezar. No es un asunto trivial. Observe que el sistema podría buscar las variables más importantes y pedir sus valores, pero lo que parece importante para él podría no serlo para usted. También podría usted especificar algún punto de partida indicando una progresión ordenada a lo largo de los distintos nodos. Pero suponga que a veces tiene usted un conjunto de valores de variables y otras veces tiene otro. En general, no tendrá valores para todas las variables, y, dependiendo de lo que tenga, el experto, obviamente, debería empezar por un punto u otro, y luego pasar a distintos puntos.

Idealmente, lo que usted querría sería poder ir al experto, darle la información que tiene y dejarle que haga lo que pueda con ella. Puede conseguirlo dándole al experto un nombre de variable y un valor para ella. El nombre de la variable determina entonces el nodo o nodos a los que el sistema debe acudir en primer lugar. Y para evitar el riesgo de equivocarse en el nombre, deberá seleccionarla del menú, o bien comprobar primero que ese nombre corresponde a una variable existente, antes de que el sistema acepte un nombre erróneo y se arme un lío.

Si teníamos el problema de dónde hacer que el experto empiece a trabajar, también tenemos el problema de dónde debe terminar.

Es muy bonito decir que el experto habrá calculado todo lo que haya

podido, de modo que siempre le dará una respuesta a su problema. En la práctica, usted no siempre estará dispuesto a permanecer sentado introduciendo datos para diversos nodos que son muy poco relevantes para el problema principal.

El método de recorrer ordenadamente los nodos permite controlar esto en alguna medida, puesto que, como el nodo N puede que contenga las decisiones finales, el objetivo será llegar al nodo N lo más rápidamente posible. En general, lo que puede hacerse es ir siempre al nodo más cercano al nodo N. Y, entonces, el problema es definir «más cercano».

La forma más sencilla es considerar cada nodo como «activo» o como «muerto». Si ya ha generado un resultado, está muerto, y no hay que acudir a él otra vez. Si no, está activo, y hay que encontrar un resultado para él. Entonces, se pueden recorrer ordenadamente los nodos, resolviendo los activos y acercándose sucesivamente al nodo N. Y en este momento es fácil darse cuenta de que el nodo N es el nodo más cercano al nodo N. Por tanto, ¿por qué no tratar de resolver este nodo el primero?. Ello ahorraría mucho trabajo.

Bien, pues si pudiera usted hacerlo, no habría necesitado antes los otros nodos. Y si no, ¿dónde habría que empezar para terminar en él?. Puede que ahora se percate usted de que la búsqueda a lo largo de los nodos no es un asunto fácil de tratar en absoluto.

El método más general consiste, como decíamos antes, en que usted proporcione los detalles de todas las variables sobre las cuales dispone de valores, y, tras introducirlos, vea el resultado.

Ahora bien, si no quiere usted introducir más detalles de los estrictamente necesarios y además tiene ciertas ideas sobre los resultados intermedios que le gustaría tener, entonces el asunto resulta ser más complicado. Suponga, por ejemplo, que usted sólo quiere que se siga el camino más rápido desde algún punto inicial hasta una solución final. ¿Qué quiere decir «más rápido»?. ¿Que se recorra el menor número posible de nodos?. ¿Que se plantee el menor número posible de preguntas?. Es usted quien debe decidir la respuesta a esa pregunta, pero con ello se demuestra que no hay una respuesta única. El optar por el menor número de preguntas le minimiza a usted el trabajo. El recorrer el mínimo de nodos le da el menor número de resultados intermedios. Seguramente, preferirá usted tener el máximo de resultados intermedios, con el mínimo esfuerzo. Es decir, recorrer todos los nodos posibles y que el sistema le haga el mínimo posible de preguntas. Y posiblemente esos objetivos sean incompatibles.

Aún arriesgándose a incrementar el esfuerzo de programación, podría usted tener siempre presente, para cada punto de la red de nodos, la distancia al nodo N en función de los nodos y de las variables aún no resueltos. Pero el incremento del esfuerzo de programación puede ser desalentador. Especialmente, si hay puntos en el programa desde los que parte más de un camino hacia el nodo N. No sólo tendrá que calcular cuál es la «distancia» por un camino, sino también cuántos caminos hay y cuál es la distancia al nodo N por todos ellos, y de algún modo tendrá que tener siempre en cuenta todos esos caminos para poder elegir el que más le guste.

De ningún modo se trata de un problema imposible. Si acude usted a la bibliografía sobre grafos y árboles, se encontrará con muchas soluciones. Pero, desde luego, no es un problema trivial.

En general, usted conseguirá reducir el esfuerzo de programación y aliviar la tensión mental, si sigue alguno de los siguientes métodos:

Ofrezca toda la información al sistema. Dé los valores de las variables que tenga y deje que el experto trabaje con todos los nodos en los que puede aplicarse esa información, hasta ver qué conclusiones obtiene.

Deje que el sistema le pida información sobre las variables que a él le parecen más importantes. Suponiendo que usted dispone de toda la información que vaya a necesitar el sistema, puede proporcionarle lo que pida y dejarle que obtenga sus conclusiones hasta que llegue a la respuesta que usted busca o hasta que le haya dado la información sobre todas las variables.

Haga que el sistema pase secuencialmente de nodo a nodo. De este modo, no se va a dejar nada, y, con el tiempo, llegará al final. Cuando usted comience a inicializar el sistema, trate de poner los nodos en un orden que le parezca razonable. Con este método, también puede usted dejar que el sistema haga cuantas deducciones pueda, porque podría suceder que llegase al final por algún camino más rápido de lo que usted creía. Y si con este procedimiento el sistema llega a algún nodo que ya ha producido un resultado, también puede saltárselo para evitarse hacer dos veces el mismo trabajo.

Es interesante señalar que puede conseguirse exactamente el mismo funcionamiento con un sistema de un solo nodo, o con un sistema que se comporte como un solo nodo. Si sólo existiera R(I,J), representando a un único nodo, podríamos tener una diversidad de resultados posibles, ...nos de los cuales podrían ser idénticos a algunas de las variables.

Si el experto decidiera sobre algún resultado y éste coincidiera con alguna variable de entrada, este resultado podría utilizarse para, por así decirlo, conectar esa variable. Tras lo cual, habría que poner una señal a ese resultado, para que sus efectos sólo se tuviesen en cuenta una sola vez.

Se podrían especificar los nombres de las variables sobre las que uno está dispuesto a dar información, o bien dejar que el sistema elija las variables sobre las que pedir información, o bien proceder secuencialmente a lo largo de toda la lista de variables.

El problema con este enfoque (hacer que todos los nodos queden contenidos en un solo gran nodo) radica en las posibilidades que deja abiertas para las dudas. Suponga, por ejemplo, que el sistema se ha destinado a predecir el tiempo, y que, para empezar, estima que hoy hace calor. ¿Hay que interrumpirlo si observamos que hoy *no* hace calor?. Al fin y al cabo, con un determinado conjunto de reglas, los *dos* resultados más probables pueden ser mutuamente exclusivos, y el sistema podría llegar a ambos resultados antes de obtener ningún otro.

Por supuesto, usted puede imaginar fácilmente un método para abordar este problema. Pero el evitar equívocos es una de las mejores formas de conseguir que los programas funcionen, y un método que en algunos aspectos dista de ser el ideal, se convierte rápidamente en el mejor de los métodos en todos los aspectos, si ocurre que es el único que puede llegar a funcionar de forma fiable.

7.4 NODOS A LA MEDIDA

En algún momento, cuando uno inicializa su sistema experto para un determinado problema, puede encontrarse con algún nodo estrambótico que parezca bien demasiado sencillo o bien demasiado complicado para dejar que aprenda mediante ejemplos.

Suponga, por ejemplo, que usted ha decidido construir un maravilloso sistema de diagnóstico médico. Una de las variables de entrada es la temperatura del paciente, y la primera cosa que quiere usted que haga el sistema es decir si la temperatura del paciente es alta o no. Ahora bien, esto es muy fácil. Si la temperatura es superior, digamos, a 37.°C, entonces cierto nodo tiene que dar resultado 1, indicando Temperatura Alta. Está claro que si usted le da suficientes ejemplos de entrenamiento al experto, éste va a aprender a obtener esa información por su cuenta. Pero parece un trabajo excesivo. Después de todo, en este caso sabe usted realmente cuáles serían las reglas. ¿Por qué perder el tiempo con una sesión de entrenamiento?

De acuerdo. Dejemos, pués, la sesión de entrenamiento y construyamos un nodo a la medida para valorar la temperatura del paciente. El problema sólo es encontrar valores de R(I,J) que den las respuestas correctas. He aquí un nodo, de dimensión R(2,2), que lo hace:

	Temperatura Alta	Temperatura Baja
Constante	0	37
Temperatura	1	0

Suponga ahora que tiene usted una entrada V(I). V(0) contiene el valor 1 (constante) y V(1) contiene la temperatura del paciente. Multiplicando por la primera regla, temperatura alta, la respuesta es siempre igual a V(1). Multiplicando por la segunda regla, temperatura baja, la respuesta es siempre igual a 37. Evidentemente, la primera regla dará el valor más alto siempre que la temperatura del paciente sea superior a 37.°C, y, así, el sistema siempre acertará en la distinción entre alta y baja temperatura. Mucho más fácil que darle un montón de ejemplos y esperar a que aprenda a encontrar la respuesta por su cuenta.

En general, la regla de decisión siempre da una respuesta que depende de:

$$D = b_1x_1 + b_2x_2 + \ldots + b_nx_n$$

de modo que, si puede usted escribir las decisiones en esta forma, todo resulta bastante sencillo.

Si no ve usted la forma de expresar en este estilo el problema, siempre puede darle algunos ejemplos al sistema y ver si éste puede calcularle algo. Si lo que quiere usted que haga el sistema es algo especialmente complicado, entonces se va a encontrar con problemas. Comentario que no es muy alentador, porque, en general, no existe ningún método para poner en esa forma todo lo que usted pudiera querer.

Suponga que tiene usted una variable de entrada que puede tomar valores entre 0 y 100, y que, si se encuentra en el margen comprendido entre 50 y 60, tiene algún significado especial, y usted quiere que el sistema experto detecte este significado. No se trata de ningún problema raro: podrían ser medidas que se toman, y los valores comprendidos entre 50 y 60 podrían ser los óptimos. Ahora bien, no es inmediato ver qué conjunto de reglas podría escribirse para detectar este margen de valores. Podría usted atacar el problema de dos formas. En primer lugar, podría usar dos nodos. El primero detectaría si el valor es superior a 50 (del mismo modo que construíamos el nodo para detectar la temperatura alta). Si el valor fuera superior a 50, el sistema podría pasar este resultado a un segundo nodo que, del mismo modo, detectaría si dicho valor

es inferior a 60. Si el sistema obtiene el resultado correcto de ambos nodos, entonces puede deducir que la variable está en el margen de 50 a 60, y el problema está resuelto. Así, con esta estructura es posible detectar los valores requeridos.

Por otra parte, parece mucho más sencillo introducir un trozo de programa:

IF V(I) >= 50 AND V(I) < = 60 THEN: Q(J) = 1:

y así para todos los casos similares, en vez de resolver el problema con una estructura bastante forzada.

Pero si empieza usted a hacer este tipo de cosas, pronto se le presentará el problema de dónde colocar elementos sueltos como éste en el conjunto global. La manera más sencilla es poner estos trozos de programa a la medida al principio del programa, de modo que siempre que el programa comience a trabajar sobre un nuevo nodo se comprueben los valores de las variables para los elementos a la medida, para ver si ya se han dado. Esto implica que esos trozos de programa van a ejecutarse más a menudo de lo que es estrictamente necesario (pero ello tiene la ventaja importante de asegurar que el experto no va a olvidar ejecutarlos).

Volviendo a nuestro maravilloso sistema de diagnóstico médico, suponga que queremos decidir si un paciente tiene fiebre. Definimos la fiebre como temperatura alta y mejillas sonrosadas (o cualquier otro conjunto de síntomas que prefiera usted elegir). El sistema ya ha decidido sobre la temperatura del paciente, de modo que todo lo que queda es ver si tiene también las mejillas sonrosadas. Dados suficientes ejemplos, el experto calculará por sí mismo un conjunto de reglas. Pero también puede usted introducirle sus propias reglas. Y éste es un ejemplo de conectivas lógicas: si (temperatura alta) y (mejillas sonrosadas)...

Adoptando otro punto de vista médico, podría estimarse que es suficiente con que el paciente tenga sólo uno de los síntomas: si (temperatura alta) o (mejillas sonrosadas)...

Las conectivas lógicas «y» y «o» son tan comunes que vale la pena tener reglas a la medida para ellas:

	Resultado 1	Resultado 2
Constante	0	n—1/2
Variable 1	1	0
Variable 2	1	0

.
.

Variable n	1	0

Con este conjunto de reglas, el resultado 1 tiene como valor máximo n, en el caso de que todas las variables tengan el valor 1. El resultado 2 tiene siempre el mismo valor, independientemente de los valores de las variables, valor que siempre es (n—1/2). Por tanto, el experto siempre elegirá el resultado 2, excepto en el caso especial de que todas las variables tengan el valor 1. Dicho de otro modo, se escogerá el resultado 1 cuando tengamos (variable 1) y (variable 2) y (variable 3) y... y (variable n).

	Resultado 1	Resultado 2
Constante	0	1/2
Variable 1	1	0
Variable 2	1	0
.		
.		
.		
Variable n	1	0

Con este otro conjunto de reglas, el resultado 2 siempre tiene el valor 1/2. El resultado 1 será inferior, con un valor 0, siempre y cuando todas las variables de entrada tengan el valor 1 en vez de 0. Con x = n tendre-resultado 1 tendrá un valor más alto que el 2. Por tanto, el experto escoge el resultado 1 siempre que tengamos (variable 1) o (variable 2) o... o (variable n).

Como puede usted observar, esos dos conjuntos de reglas son muy parecidos entre sí. De hecho, si a la constante le damos el valor (x—1/2) para el resultado 2, el experto dará el resultado 1, en cuanto x o más de las variables de entrada tengan el valor 1 en vez de 0. Con x = n tendremos la conectiva «y», y con x = 1 la conectiva «o». Y usted puede poner cualquier otro número, si ello conviene para su aplicación.

La ventaja de tener sus propias reglas construídas de ese modo, es que usted siempre sabe exactamente qué es lo que va a hacer el sistema en cada momento (o, en cualquier caso, aproximadamente). La ventaja de que el sistema aprenda sus propias reglas a través de ejemplos, es que con ello se tiene una forma fácil de introducir las reglas en el sistema, con el valor añadido de que usted no tiene que molestarse en sentarse y calcular previamente cómo han de ser tales reglas.

7.5 INSTRUCCIONES ESPECIFICAS

Si uno piensa en sistemas maravillosos para diagnóstico médico, reflexionará sobre el asunto surgido en el apartado anterior, el asunto de la fiebre. Supongamos que consideramos que la fiebre consiste en temperatura alta y mejillas sonrosadas. Ya hemos dicho cómo se puede introducir esto en nuestra base de conocimientos en forma de reglas, ya sea explícitamente o por medio de ejemplos, y cómo puede entonces diagnosticarse la fiebre. Pero, ¿no sería mucho más fácil escribir simplemente la línea de BASIC:

```
IF T > 37 AND F = 1 THEN: PRINT «FIEBRE»
```

donde el valor de T es la temperatura del paciente, y F = 1, si tiene las mejillas sonrosadas?

Bien, pues sí, ciertamente sería mucho más sencillo y poco susceptible de errores.

De hecho, usted podría tener toda una colección de sentencias como ésa, que constituirían un sistema con el que diagnosticar cualquier cosa (siempre y cuando su programa cubriera esa «cualquier cosa»). Sería mucho más rápido y fácil hacer las cosas así, porque el BASIC es un lenguaje de programación tan sencillo que, con un poco de práctica, usted puede sentarse y escribir el programa casi tan rápidamente como el escribir las reglas en un sistema de propósito más general. Hasta que el programa no estuviese completamente definido, podría ser algo errático, pero un enfoque con un sistema de propósito general también es algo errático hasta que no es seguro que cubre todas las posibilidades.

No obstante, hay que tener en cuenta una afirmación teórica que a veces se hace sobre los sistemas expertos: que, idealmente, son programas de propósito general. Contienen un método para establecer juicios expertos en general. Y se supone que, teniendo un sistema experto, basta añadir una base de conocimientos para obtener otro experto diferente. Y esto es algo que usted no puede conseguir si se sienta y escribe su propio programa para un problema específico: si escribe usted mediante este método un sistema para diagnóstico médico, no podrá luego modificarlo fácilmente para que resuelva otro problema.

Todo ello no constituye ninguna crítica seria, si es eso lo que quiere usted hacer. Después de todo, la mayoría de los sistemas expertos sólo abordan realmente un problema específico, no son verdaderos sistemas de propósito general. El que usted está construyendo podría ser de propósito general, capaz de hacer cualquier cosa. Pero no hay muchos

otros como él. Y, como hemos visto anteriormente, a veces tiene sus limitaciones.

7.6 CÓMO SALVAGUARDAR SU EXPERTO

Se acaba el día. Sentado ante el teclado, usted ha estado entrenando a su experto hasta que ha conseguido dotarle de un buen grado de habilidad, y es hora de irse a la cama. Desconecta la máquina y ¡visto y no visto!: la experiencia desaparece como por arte de magia.

Ya podía usted haber imaginado que iba a suceder tal cosa: después de todo, sólo eran datos. Pero tener un experto que hay que reconstruir desde el principio cada vez que se quiere hacerlo funcionar no sirve para nada. Queremos que vaya a mejor cada día, no a peor.

El programa puede salvaguardarse (con la sentencia SAVE) en cassette o en disco, pero la forma en que pueden salvaguardarse los datos depende del sistema operativo que tenga usted. Si tiene un sistema basado en disco, habría sido mejor utilizar técnicas de manejo de ficheros que todos aquellos vectores y matrices (¡a buena hora me lo dice!, exclamará usted), pero si no tiene discos, y es posible que no los tenga, no puede hacerlo. Además, el número y el tipo de ficheros que pueden abrirse depende, en gran medida, del sistema.

El procedimiento más sencillo es volcar toda la memoria de la máquina en cassette o en disco, según lo que tenga. De nuevo, depende de su sistema el que esto funcione o no. Antes de empezar a construir laboriosamente el conocimiento del experto, ensaye el volcado de la memoria. Hágalo en mitad de la ejecución de un programa, utilizando cualquier método que le sugiera su manual que podría funcionar. Si todo va bien, deberá poder desconectar la máquina, volverla a conectar, cargar la memoria y seguir desde el mismo punto en que estaba. Si funciona así, este método está muy bien. Si no, vea si puede conseguir, de algún otro modo, que el programa vuelva a seguir ejecutándose, quizás desde un punto distinto a aquél en que se interrumpió. Pruebe con una sentencia GOTO mejor que con una RUN, para entrar en el programa.

Si, a pesar de todo, no funciona, usted tendrá que pensar en alguna otra cosa. Y esa otra cosa es probable que consista en seleccionar vectores y matrices específicos y salvaguardarlos. Por ejemplo, R(I,J) para las reglas, y M(2,J) para los valores mínimos y máximos de las variables.

Lo que usted puede hacer es escribir una pequeña rutina que salvaguarde el programa y todas las variables dimensionadas que quiera. Al

final del día, ejecutaría esta ruina, que, por ejemplo, para el Apple II sería:

SAVE: STORE R: STORE M y así sucesivamente.

Esta rutina depositaría en cassette o en disco el programa y las variables en el orden escrito.

Luego, cuando arranque usted al día siguiente, cargue (LOAD) el programa e inmediatamente teclee RUN. Pero cuide de que las primeras sentencias de su programa sean RECALL, en el mismo orden en que tenía las STORE. Con ello, se recogerá la información de la cassette o del disco en el mismo orden en que se almacenó.

En el Sinclair Spectrum puede salvaguardarse el programa completo con todos los datos, incluyendo vectores y matrices, con una sentencia del tipo

SAVE «EXPERT»

Pero, cuando vuelva a cargar el programa, no debe teclear RUN, porque con ello se pondrían a cero todas las variables. En su lugar, teclee, por ejemplo, GOTO 10, donde 10 es un número de línea. Esta técnica es la que se utiliza en el listado del Apartado 7.7.

7.7 EL CÓDIGO MULTINODO

Para evitarle el inconveniente de tener que hacer nada, he aquí un sistema experto por aprendizaje completo, con menú, multinodo, que le permitirá, después de algunos segundos de práctica, alcanzar objetivos razonablemente modestos. Como, por ejemplo, la Dominación del Mundo (siempre y cuando pueda usted pensar en algunos ejemplos para darle).

Fig. 7.3
Experto multinodo con menú

(a) Listado para el Apple II

```
10 Z = 0:W = 1:TW = 2:MI =  - 1E38:T
   H = 13:TE = 10:FO = 40
20 GOSUB 1200
30 PRINT : PRINT : PRINT "1. INICIA
```

```
    LIZAR EL EXPERTO."
40  PRINT "2. PONER EJEMPLOS."
50  PRINT "3. EJERCITAR AL EXPERTO."

60  PRINT "4. SESION DE ENTRENAMIENT
    O."
70  PRINT "5. EJECUCION NORMAL."
80  PRINT "6. SALVAR EL EXPERTO ACTU
    AL.": PRINT "7. EXAMINAR LOS VAL
    ORES ORDENADOS Y LOS EJEMPLOS."
90  PRINT : PRINT : INPUT "SELECCION
    E OPCION ";OP
100  IF OP = W THEN 170
110  IF OP = TW THEN 680
120  IF OP = 3 THEN 970
130  IF OP = 4 THEN T$ = "S": GOTO 3
    30
140  IF OP = 5 THEN T$ = "N": GOTO 3
    20
150  IF OP = 6 THEN 1100
160  IF OP = 7 THEN  GOSUB 1120: GOTO
    20
170  GOSUB 1200: PRINT "PARA OBTENER
     UN SISTEMA EXPERTO.": PRINT "CO
    NTESTA A LAS SIGUIENTES PREGUNTA
    S: ": PRINT : PRINT : INPUT "CUA
    NTOS NODOS TIENE? ";N
180  PRINT : PRINT : PRINT "CUAL ES
    EL MAXIMO": PRINT "NUMERO DE VAR
    IABLES": PRINT "PARA CADA NODO?
    ": INPUT MV
190  PRINT : PRINT : PRINT "CUAL ES
    EL MAXIMO": PRINT "NUMERO DE RES
    ULTADOS": PRINT "PARA CADA NODO?
     ": INPUT MQ
200  DIM V$(MV,N),V(MV,N),VC(MV,N),R
    V(MV,N),M(TW,MV,N),E(MV + W,50,N
    ),VA(N),QU(N),Q$(MQ,N),R(MV,MQ,N
    ),D(MQ),PD(MQ),QC(MQ,N),EZ(N)
210  FOR NO = W TO N: GOSUB 1200: PRINT
    "NODO ";NO: PRINT : PRINT "CUANT
```

```
    AS VARIABLES TIENES": INPUT "EN
    ESTE NODO? ";VA(NO)
220  PRINT : PRINT "NOMBRE ESTAS VAR
    IABLES:": PRINT
230  FOR I = W TO VA(NO): VTAB (TH):
     FOR P = W TO TE: PRINT  SPC( FO
    );: NEXT P: VTAB (TH): PRINT : PRINT
    "LA VARIABLE ";I;" ES ";: INPUT
    V$(I,NO):X$ = "N"
240  FOR P = W TO NO: FOR Q = W TO V
    A(P): IF V$(I,NO) = V$(Q,P) AND
    NO <  > P THEN  PRINT "(";V$(I,N
    O);") YA SUCEDE": PRINT "EN EL N
    ODO ";P: PRINT "CON VALOR MINIMO
    = ";M(W,Q,P): PRINT "Y MAXIMO= "
    ;M(TW,Q,P): INPUT "SUCEDE AQUI I
    GUAL? ";X$
250  IF X$ = "S" THEN M(W,I,NO) = M(
    W,Q,P):M(TW,I,NO) = M(TW,Q,P):X$
     = "N": GOTO 280
260  NEXT Q: NEXT P
270  PRINT : PRINT : PRINT "VALOR MI
    NIMO= ";: INPUT M(W,I,NO): PRINT
    "VALOR MAXIMO= ";: INPUT M(TW,I,
    NO)


280  NEXT I
290  GOSUB 1200: PRINT "NODO ";NO: PRINT
    "CUANTOS RESULTADOS TIENES": INPUT
    "EN ESTE NODO? ";QU(NO)
300  PRINT : PRINT "NOMBRA ESTOS RES
    ULTADOS:": PRINT : FOR I = W TO
    QU(NO): PRINT "EL RESULTADO ";I;
    " ES ";: INPUT Q$(I,NO): NEXT I
310  NEXT NO: GOTO 20
320  IF T$ = "N" THEN  GOSUB 1200: PRINT
    "EXPERTO PREPARADO": PRINT : PRINT
    : PRINT "VOY A PEDIRTE INFORMACI
    ON": PRINT "E INTENTARE HACER UN
    A DEDUCCION CORRECTA": GOTO 350
```

```
330  GOSUB 1200: PRINT "SESION DE EN
     TRENAMIENTO": PRINT : PRINT : PRINT
     "VOY A PEDIRTE QUE METAS UNAS VA
     RIABLES": PRINT "Y TE DARE UN RE
     SULTADO": PRINT "DEBES DECIRME S
     I ES CORRECTO O NO"
340  PRINT : PRINT "PULSA CUALQUIER
     TECLA PARA SEGUIR": GET X$
350  FOR NO = W TO N: FOR I = W TO V
     A(NO):VC(I,NO) = W:V(I,NO) = Z: NEXT
     I: FOR J = W TO QU(NO):QC(J,NO) =
     W: NEXT J: NEXT NO: FOR NO = W TO
     N:V = VA(NO):Q = QU(NO): FOR J =
     W TO Q:D(J) = Z
360  FOR K = W TO V: IF VC(K,NO) = Z
      THEN D(J) = D(J) + V(K,NO) * R(
     K,J,NO)
370  NEXT K: NEXT J: FOR J = W TO Q:
     PD(J) = D(J): NEXT J
380  GOSUB 1200: PRINT "NODO ";NO
390 H = Z:VV = Z: FOR I = W TO V:RV(
     I,NO) = Z:M = Z:N1 = Z: FOR J =
     W TO Q:M = M + R(I,J,NO) * VC(I,
     NO) * QC(J,NO):N1 = N1 + QC(J,NO
     ): NEXT J: IF N1 THEN M = M / N1
400  FOR J = W TO Q:RV(I,NO) = RV(I,
     NO) +  ABS (M(W,I,NO) - M(TW,I,N
     O)) * ((R(I,J,NO) - M) ^ TW) * V
     C(I,NO) * QC(J,NO): NEXT J: IF R
     V(I,NO) > H THEN VV = I:H = RV(I
     ,NO)
410  NEXT I: IF H = Z THEN 560
420 VC(VV,NO) = Z
430  FOR J = W TO Q - W: FOR K = J +
     W TO Q: IF R(VV,J,NO) <  > R(VV,
     K,NO) AND QC(J,NO) AND QC(K,NO) THEN
     450
440  NEXT K: NEXT J: GOTO 390
450  PRINT VV;". ";V$(VV,NO);: INPUT
     V(VV,NO): IF V(VV,NO) < M(W,VV,N
     O) OR V(VV,NO) > M(TW,VV,NO) THEN
```

```
     PRINT : PRINT "VALOR FUERA DE E
    SCALA": PRINT "HAS DADO VALOR MI
    NIMO= ";M(W,VV,NO): PRINT "Y VAL
    OR MAXIMO= ";M(TW,VV,NO): GOTO 4
    50
460  FOR P = W TO N: FOR V1 = W TO V
    A(P): IF V$(V1,P) = V$(VV,NO) THEN
    V(V1,P) = V(VV,NO):VC(V1,P) = Z
470  NEXT V1: NEXT P:D = MI: FOR J =
    W TO Q:D(J) = D(J) + V(VV,NO) *
    R(VV,J,NO):PD(J) = D(J): IF D(J)
     >  = D THEN D = D(J):HJ = J
480  NEXT J
490  FOR I = W TO V: FOR J = W TO Q
500  IF VC(I,NO) = W THEN  IF R(I,J,
    NO) > R(I,HJ,NO) THEN PD(J) = PD
    (J) + (R(I,J,NO) - R(I,HJ,NO)) *
    M(TW,I,NO)
510  IF VC(I,NO) = W THEN  IF R(I,J,
    NO) < R(I,HJ,NO) THEN PD(J) = PD
    (J) + (R(I,J,NO) - R(I,HJ,NO)) *
    M(W,I,NO)
520  NEXT J: NEXT I
530 H2 = MI: FOR J = W TO Q: IF PD(J
    ) >  = H2 THEN H2 = PD(J):HI = J

540  IF PD(J) < PD(HJ) AND J <  > HJ
     THEN QC(J,NO) = Z
550  NEXT J: IF PD(HI) <  > PD(HJ) THEN
    390
560  IF HJ AND T$ = "S" THEN  PRINT
    : PRINT "DEDUZCO EL RESULTADO ";
    Q$(HJ,NO): INPUT A$: IF A$ = "S"
     THEN  GOSUB 780: GOSUB 850: NEXT
    NO: PRINT : INPUT "QUIERES QUE C
    ONTINUE ENTRENANDOME? ";B$: IF B
    $ = "S" THEN 350
570  IF T$ = "N" THEN  PRINT "SUGIER
    O EL RESULTADO ";Q$(HJ,NO): GOSUB
    780: NEXT NO: PRINT : INPUT "QUI
    ERES QUE CONTINUE? ";B$: IF B$ =
```

```
    "S" THEN 350
580  IF (A$ = "S" AND B$ = "N" AND T
    $ = "S") OR (B$ = "N" AND T$ = "
    N") THEN 20
590  IF HJ = Z THEN 600
600  FOR I = W TO Q: PRINT I;". ";Q$
    (I,NO): NEXT I: PRINT : INPUT "C
    UAL ES EL RESULTADO? ";HI
610  FOR I = W TO V: IF VC(I,NO) THEN
     PRINT "QUE VALOR TENIA ";V$(I,N
    O);"? ";: INPUT V(I,NO):VC(I,NO)
     = Z
620  NEXT I
630  FOR I = W TO Q: IF D(I) >  = D(
    HI) AND I <  > HI THEN  FOR J =
    W TO V:R(J,I,NO) = R(J,I,NO) - V
    (J,NO): NEXT J
640  NEXT I
650  FOR J = W TO V:R(J,HI,NO) = R(J
    ,HI,NO) + V(J,NO): NEXT J:HJ = H
    I: GOSUB 780: GOSUB 850: NEXT NO

660  PRINT : PRINT : INPUT "QUIERES
    QUE CONTINUE ENTRENANDOME? ";C$:
     IF C$ = "S" THEN 330
670  GOTO 20
680  FOR NO = W TO N: GOSUB 1200: PRINT
    "NUDO ";NO: PRINT : PRINT : PRINT
    "PONER ALGUNOS EJEMPLOS": PRINT
    : PRINT : PRINT "CUANTOS EJEMPLO
    S TIENES": INPUT "EN ESTE NODO?
    ";N2
690  FOR I = W TO N2
700  GOSUB 1200: PRINT "EJEMPLO NO.
    ";I" DEL NODO ";NO: PRINT : PRINT

710  FOR J = W TO VA(NO)
720  PRINT "LA VARIABLE ";J;" (";V$(
    J,NO);") VALE ";: INPUT E(J,I +
    EZ(NO),NO)
730  NEXT J
```

```
740  PRINT : PRINT "Y EL RESULTADO E
     S: ": FOR J = W TO QU(NO): PRINT
     J;". ";Q$(J,NO): NEXT J: INPUT "
     RESPONDE CON UN NUMERO ";E(VA(NO
     ) + W,I + EZ(NO),NO)
750  NEXT I
760 EZ(NO) = EZ(NO) + N2
770  NEXT NO: GOTO 20
780  FOR P = W TO N
790  FOR V1 = W TO VA(P)
800  IF V$(V1,P) = Q$(HJ,NO) THEN V(
     V1,P) = M(TW,V1,P):VC(V1,P) = Z
810  FOR R = W TO Q: IF V$(V1,P) = Q
     $(R,NO) AND R <  > HJ THEN VC(V1
     ,P) = Z
820  NEXT R
830  NEXT V1: NEXT P
840  RETURN
850  INPUT "DESEAS GUARDAR ESTE EJEM
     PLO? ";E$: IF E$ = "N" THEN  RETURN

860  FOR J = W TO VA(NO): IF VC(J,NO
     ) = Z THEN 900
870 VV = J:VC(J,NO) = Z
880  PRINT "QUE VALOR TENIA:"
890  PRINT VV;". ";V$(VV,NO);: INPUT
     V(VV,NO): FOR P = W TO N: FOR Q =
     W TO VA(P): IF V$(Q,P) = V$(VV,N
     O) THEN V(Q,P) = V(VV,NO):VC(Q,P
     ) = Z
900  NEXT J
910 EZ(NO) = EZ(NO) + W
920  FOR J = W TO VA(NO)
930 E(J,EZ(NO),NO) = V(J,NO)
940  NEXT J
950 E(VA(NO) + W,EZ(NO),NO) = HJ
960  RETURN
970  GOSUB 1200: PRINT "EL EXPERTO E
     STA TRABAJANDO": FOR NO = W TO N
     :DE = EZ(NO):I = Z: FOR P = W TO
     DE * 200:I =  INT ( RND (W) * DE + W)
```

```
980 D = Z:C = Z:V = VA(NO):Q = QU(NO
    )
990  FOR J = W TO V
1000 D = D + E(J,I,NO) * R(J,E(V + W
    ,I,NO),NO)
1010  NEXT J
1020  FOR K = W TO Q
1030 D2 = Z
1040  IF K < > E(V + W,I,NO) THEN  FOR
    J = W TO V:D2 = D2 + E(J,I,NO) *
    R(J,K,NO): NEXT J
1050  IF D2 > = D AND K < > E(V +
    W,I,NO) THEN C = C + W: FOR J =
    W TO V:R(J,K,NO) = R(J,K,NO) - E
    (J,I,NO): NEXT J
1060  IF C = W THEN  FOR J = W TO V:
    R(J,E(V + W,I,NO),NO) = R(J,E(V +
    W,I,NO),NO) + E(J,I,NO): NEXT J:
    C = C + W
1070  NEXT K
1080  NEXT P
1090  NEXT NO: GOTO 20
1100  GOSUB 1200: PRINT "SALVAR EL E
    XPERTO ACTUAL": PRINT : PRINT "C
    ALL-151": PRINT : PRINT "*0000.C
    000W": PRINT : PRINT "PARA CASSE
    TTE": PRINT : PRINT "PARA REGRES
    AR": PRINT "*0000.C000R"
1110  PRINT : PRINT : PRINT "PULSA '
    CONT' PARA CONTINUAR": STOP : GOTO
    20
1120  FOR PP = 1 TO N
1130  PRINT "NUDO ";PP
1140  PRINT "E(I,J,NODE)"
1150  FOR QQ = W TO VA(PP): FOR RR =
    W TO QU(PP): PRINT E(QQ,RR,PP);"
     ";: NEXT RR: PRINT : NEXT QQ
1160  PRINT "R(I,J,NODE)"
1170  FOR QQ = W TO VA(PP): FOR RR =
    W TO QU(PP): PRINT R(QQ,RR,PP);"
     ";: NEXT RR: PRINT : NEXT QQ
```

```
1180  GET A$
1190  NEXT PP
1200  HOME : PRINT ,"EXPERTO": PRINT
    ,"---------": PRINT : PRINT : PRINT
    : RETURN : REM  CABECERA
```

(b) Listado para el Sinclair Spectrum

```
  10 LET mi=-1e38: LET hj=0: LET a$="": LE
T b$="": LET t$="": LET d=0
  20 GO SUB 1200
  40 PRINT '''"1. Inicializacion del Exper
to"'"2. Entrada de ejemplos"'"3. Ejercicio
 con el Experto"'"4. Sesion de entrenamien
to"'"5. Ejecucion normal"'"6. Guardar el E
xperto en curso"'"7. Visualizacion de Regl
as y        Ejemplos"'"8. PARAR"
 100 INPUT "Elige una opcion: ";op
 110 IF op=1 THEN  GO TO 170
 120 IF op=2 THEN  GO TO 670
 130 IF op=3 THEN  GO TO 970
 140 IF op=4 THEN  LET t$="s": GO TO 330
 150 IF op=5 THEN  LET t$="n": GO TO 320
 160 IF op=6 THEN  GO TO 1100
 165 IF op=7 THEN  GO TO 1120
 166 IF op=8 THEN  STOP
 168 GO TO 20
 170 GO SUB 1200: PRINT "Crear un Sistema
Experto"'"Por favor, responde a lo siguien
te:": INPUT "Cuantos nodos tienes?";n
 180 PRINT '"Cual es el numero maximo"'"d
e variables en cualquier nodo?": INPUT mv
 190 PRINT '"Cual es el numero maximo"'"d
e resultados en cualquier nodo?": INPUT mq
 200 DIM v$(mv,n,20): DIM v(mv,n): DIM c(m
v,n): DIM r(mv,n): DIM m(2,mv,n): DIM e(mv
+1,50,n): DIM a(n): DIM u(n): DIM q$(mq,n,
20): DIM s(mv,mq,n): DIM d(mq): DIM p(mq):
 DIM q(mq,n): DIM z(n)
 210 FOR o=1 TO n: GO SUB 1200: PRINT AT 2
0,0;"En el nodo ";(o): INPUT "Cuantas vari
ables hay?";a(o)
 220 PRINT '"Por favor, nombra estas varia
bles"
 230 FOR i=1 TO a(o): INPUT "La variable "
;(i);" es ";v$(i,o): LET x$="n"
```

```
240 FOR p=1 TO 0: FOR q=1 TO a(p): IF v$(

i,o)=v$(q,p) AND o<>p THEN  PRINT (v$(i,o)
);'"ya ocurre en el nodo ";p'"con valor mi
nimo "(m(1,q,p))'"y valor maximo"'(m(2,q,p
)): INPUT "Es igual aqui?";x$
 250 IF x$="s" THEN  LET m(1,i,o)=m(1,q,p)
: LET m(2,i,o)=m(2,q,p): LET x$="n": GO TO
 280
 260 NEXT q: NEXT p
 270 INPUT '"Tiene un valor minimo=";m(1,i
,o): INPUT "Y un valor maximo=";m(2,i,o)
 280 NEXT i
 290 GO SUB 1200: INPUT "Cuantos resultado
s "'"en el nodo";(o);"?";u(o)
 300 PRINT '"Por favor nombra estos result
ados:": FOR i=1 TO u(o): INPUT "El resulta
do ";(i);" es ";q$(i,o): NEXT i
 310 NEXT o: GO TO 20
 320 IF t$="n" THEN  GO SUB 1200: PRINT "E
xperto Preparado"'"Te pedire informacion"
'"y tratare de hacer una deducion"'"correc
ta": GO TO 350
 330 GO SUB 1200: PRINT "Sesion de Entrena
miento"'"Te pedire que introduzcas alguna
s"'"variables, luego hare"'"una deducion.
Debes decirme"'"si es correcta."
 340 PRINT '"Pulsa cualquier tecla para co
ntinuar"
 345 LET a$=INKEY$: IF a$="" THEN  GO TO 3
45
 350 FOR o=1 TO n: FOR i=1 TO a(o): LET c(
i,o)=1: LET v(i,o)=0: NEXT i: FOR j=1 TO u
(o): LET q(j,o)=1: NEXT j: NEXT o: FOR o=1
 TO n: LET v=a(o): LET q=u(o): FOR j=1 TO
q: LET d(j)=0
 360 FOR k=1 TO v: IF c(k,o)=0 THEN  LET d
(j)=d(j)+v(k,o)*s(k,j,o)
 370 NEXT k: NEXT j: FOR j=1 TO q: LET p(j
)=d(j): NEXT j
 380 GO SUB 1200: PRINT "Nodo ";o
```

```
 390 LET h=0: LET vv=0: FOR i=1 TO v: LET
r(i,o)=0: LET m=0: LET n1=0: FOR j=1 TO q:
 LET m=m+s(i,j,o)*c(i,o)*q(j,o): LET n1=n1
+q(j,o): NEXT j: IF n1<>0 THEN  LET m=m/n1
 400 FOR j=1 TO q: LET r(i,o)=r(i,o)+ABS (
m(1,i,o)-m(2,i,o))*(ABS (s(i,j,o)-m)^2)*c(
i,o)*q(j,o): NEXT j: IF r(i,o)>h THEN  LET
 vv=i: LET h=r(i,o)
 410 NEXT i: IF h=0 THEN  GO TO 550
 415 LET c(vv,o)=0
 420 FOR j=1 TO q-1: FOR k=j+1 TO q: IF s(
vv,j,o)<>s(vv,k,o) AND q(j,o)<>0 AND q(k,o
)<>0 THEN  GO TO 440
 430 NEXT k: NEXT j: GO TO 390
 440 INPUT (vv);". ";(v$(vv,o));v(vv,o): I
F v(vv,o)<m(1,vv,o) OR v(vv,o)>m(2,vv,o) T
HEN  PRINT ''"La entrada esta fuera del ma
rgen"'"El minimo era ";(m(1,vv,o))'"y el m
aximo ";(m(2,vv,o)): GO TO 440
 450 FOR p=1 TO n: FOR v=1 TO a(p): IF v$(
u,p)=v$(vv,o) THEN  LET v(u,p)=v(vv,o): LE
T c(u,p)=0
 460 NEXT u: NEXT p: LET d=mi: FOR j=1 TO
q: LET d(j)=d(j)+v(vv,o)*s(vv,j,o): LET p(
j)=d(j): IF d(j)>=d THEN  LET d=d(j): LET
hj=j
 470 NEXT j
 480 FOR i=1 TO v: FOR j=1 TO q
 490 IF c(i,o)=1 THEN  IF s(i,j,o)>s(i,hj,
o) THEN  LET p(j)=p(j)+(s(i,j,o)-s(i,hj,o)
)*m(2,i,o)
 500 IF c(i,o)=1 THEN  IF s(i,j,o)<s(i,hj,
o) THEN  LET p(j)=p(j)+(s(i,j,o)-s(i,hj,o)
)*m(1,i,o)
 510 NEXT j: NEXT i
 520 LET h2=mi: FOR j=1 TO q: IF p(j)>=h2
THEN  LET h2=p(j): LET hi=j
 530 IF p(j)<p(hj) AND j<>hj THEN  LET q(j
,o)=0
 540 NEXT j: IF p(hi)>p(hj) THEN  GO TO 39
0
```

```
 550 IF hj<>0 AND t$="s" THEN  INPUT "Pued
o deducir un resultado:"'(q$(hj,o));a$: IF
 a$="s" THEN  GO SUB 770: GO SUB 840: NEXT
 o: INPUT "Continua el entrenamiento?";b$:
 IF b$="s" THEN  GO TO 350
 560 IF t$="n" THEN  PRINT "Sugiero ";(q$(
hj,o))'"como probable.": GO SUB 770: NEXT
o: INPUT "Quieres continuar?";b$: IF b$="s
" THEN  GO TO 350
 570 IF (a$="s" AND b$="n" AND t$="s") OR
(b$="n" AND t$="n") THEN  GO TO 20
 590 FOR i=1 TO q: PRINT i;". ";(q$(i,o)):
 NEXT i: INPUT "Que resultado es?";hi
 600 FOR i=1 TO v: IF c(i,o)>0 THEN  INPUT
 "Que valor era ";(v$(i,o));" ?";v(i,o): L
ET c(i,o)=0
 610 NEXT i
 620 FOR i=1 TO q: IF d(i)>=d(hi) AND i<>h
i THEN  FOR j=1 TO v: LET s(j,i,o)=s(j,i,o
)-v(j,o): NEXT j
 630 NEXT i
 640 FOR j=1 TO v: LET s(j,hi,o)=s(j,hi,o)
+v(j,o): NEXT j: LET hj=hi: GO SUB 770: GO
 SUB 840: NEXT o
 650 INPUT "Continua el entrenamiento?";c$
: IF c$="s" THEN  GO TO 330
 660 GO TO 20
 670 FOR o=1 TO n: GO SUB 1200: PRINT "Nod
o ";o'"Pon algunos ejemplos"'''"Cuantos ej
emplos en este nodo?": INPUT n2
 680 FOR i=1 TO n2
 690 GO SUB 1200: PRINT "Ejemplo numero ";
(i);'"en el nodo ";(o)''
 700 FOR j=1 TO a(o)
 710 INPUT "La variable ";(j)'"(";(v$(j,o)
);") es ";e(j,i+z(o),o)
 720 NEXT j
 730 PRINT ''"Y el resultado es?": FOR j=1
 TO u(o): PRINT j;".";q$(j,o): NEXT j: INP
UT "Por favor, entra por Numero";e(a(o)+1,
i+z(o),o)
```

```
 740 NEXT i
 750 LET z(o)=z(o)+n2
 760 NEXT o: GO TO 20
 770 FOR p=1 TO n
 780 FOR y=1 TO a(p)
 790 IF v$(y,p)=q$(hj,o) THEN  LET v(y,p)=
m(2,y,p): LET c(y,p)=0
 800 FOR r=1 TO q: IF v$(y,p)=q$(r,o) AND
r<>hj THEN  LET c(y,p)=0
 810 NEXT r
 820 NEXT y: NEXT p
 830 RETURN
 840 INPUT "Guardamos este ejemplo?";e$: I
F e$="n" THEN  RETURN
 850 FOR j=1 TO a(o): IF c(j,o)=0 THEN  GO
 TO 890
 860 LET vv=j: LET c(i,j)=0
 870 INPUT "Que valor era:"'(v$(vv,o));v(v
v,o): FOR p=1 TO n: FOR q=1 TO a(p): IF v$
(q,p)=v$(vv,o) THEN  LET v(q,p)=v(vv,o): L
ET c(q,p)=0
 890 NEXT i
 900 LET z(o)=z(o)+1
 910 FOR j=1 TO a(o)
 920 LET e(j,z(o),o)=v(j,o)
 930 NEXT j
 950 LET e(a(o)+1,z(o),o)=hj
 960 RETURN
 970 GO SUB 1200: PRINT '"El Experto esta
trabajando en ello": FOR o=1 TO n: LET de=
z(o): LET i=0: FOR p=1 TO de*200: LET i=IN
T (1+de*RND)
 980 LET d=0: LET c=0: LET v=a(o): LET q=u
(o)
 990 FOR j=1 TO v
1000 LET d=d+e(i,j,o)*s(j,e(v+1,i,o),o)
1010 NEXT j
1020 FOR k=1 TO q
1030 LET d2=0
1040 IF k<>e(v+1,i,o) THEN  FOR j=1 TO v:
LET d2=d2+e(j,i,o)*s(j,k,o): NEXT j
```

```
1050 IF d2>=d AND k<>e(v+1,i,o) THEN  LET
c=c+1: FOR j=1 TO v: LET s(j,k,o)=s(j,k,o)
-e(j,i,o): NEXT j
1060 IF c=1 THEN  FOR j=1 TO v: LET s(j,e(
v+1,i,o),o)=s(j,e(v+1,i,o),o)+e(j,i,o): NE
XT j: LET c=c+1
1070 NEXT k
1080 NEXT p
1090 NEXT o: GO TO 20
1100 GO SUB 1200: PRINT ''"Guarda el Exper
to en curso"'"(Tanto programa como datos)"
'"Por favor, inserta el cassete"
1110 SAVE "multinodo"
1115 PRINT '"Guardado bajo el nombre 'mult
inodo'"
1116 PRINT "Pulsa cualquier tecla para con
tinuar"
1117 LET a$=INKEY$: IF a$="" THEN  GO TO 1
117
1118 GO TO 20
1120 CLS : FOR x=1 TO n
1130 PRINT "Nodo ";x
1140 PRINT "e(i,j,nodo)"
1150 FOR y=1 TO a(x)+1: FOR z=1 TO z(x): P
RINT e(y,z,x);" ";: NEXT z: PRINT : NEXT y
1155 PRINT "Pulsa cualquier tecla para con
tinuar"
1156 LET a$=INKEY$: IF a$="" THEN  GO TO 1
156
1160 CLS : PRINT "r(i,j,nodo)"
1170 FOR y=1 TO a(x): FOR z=1 TO u(x): PRI
NT s(y,z,x);" ";: NEXT z: PRINT : NEXT y
1175 PRINT "Pulsa cualquier tecla para con
tinuar"
1176>LET a$=INKEY$:IF a$=""THEN GO TO 1176
1180 NEXT x
1190 PRINT "Pulsa cualquier tecla para con
tinuar"
1196 LET a$=INKEY$: IF a$="" THEN  GO TO 1
196
1198 GO TO 20
```

```
1200 CLS : PRINT ,"EXPERTO""_______"":
RETURN
```

7.8 ALGUNOS EJEMPLOS

Si verdaderamente usted ha conseguido teclear todo ese programa, entonces lo primero que querrá hacer será retirarse en soledad a algún sitio discreto y cuidar de sus destrozados dedos durante una semana o así.

Pero, una vez hecho eso, puede ocurrir que quiera usted tratar de ejecutarlo, en cuyo caso algunos ejemplos vendrán bien para que pueda saber si funciona o no.

Por tanto, teclee RUN y se encontrará con un menú de siete elementos. Escoja la primera opción para inicializar al invento. Dígale que tiene un nodo y que el número máximo de variables y de resultados es dos.

Llame a las variables con los nombres de Alas (con valor mínimo 0 y valor máximo 1) y Motor (mínimo 0, máximo 1). Déle a los resultados los nombres de Pájaro y Avión.

Ahora se volverá a encontrar con el menú, y puede hacer una de dos cosas: o bien ir a una sesión de entrenamiento y hacerle aprender lentamente, o bien ir a la opción de Introducir Ejemplos para darle un lote de ejemplos con los que se ponga a trabajar. Vayamos a esta segunda opción para hacer las cosas más fáciles. Déle dos ejemplos. El sistema preguntará sobre las variables y los resultados, de modo que, como primer ejemplo, puede darle un pájaro, y como segundo, un avión.

Vaya entonces a la opción de Ejercitar Experto y descanse durante uno o dos minutos mientras el experto calcula sus reglas.

Cuando el experto vuelva, entre en la opción siete para echar un vistazo a las reglas que ha desarrollado. La matriz E debe contener los ejemplos que usted ha dado, del siguiente modo:

1	1	esta línea corresponde a Alas: Sí en ambos casos
0	1	esta línea es la de Motor: sólo existe en el segundo ejemplo
1	2	esta línea es la de Resultados: 1 para Pájaro y 2 para Avión.

Y la matriz R debe estar así:

1	-1	esta línea corresponde a Alas
-2	2	esta línea corresponde a Motor

Eche un vistazo a estas reglas y convénzase de que separan los resultados, tal como deben de hacerlo. Una vez hecho eso, podrá quedar más tranquilo sobre la posibilidad de que el programa, como un todo, funcione, si bien es cierto que no hay necesidad de hacer siempre tal cosa.

Pulse cualquier tecla para volver al menú.

Tome la opción 5: Funcionamiento normal.

El programa empezará preguntando por Motor. Observe que ésta parece ser la variable más importante en la matriz R (y realmente lo es). Responda 1, y el experto dirá que es un Avión sin preguntar nada más.

Responda S para seguir.

El sistema pregunta de nuevo por Motor. Conteste 0. Luego pregunta por Alas. Si responde usted con 1, el sistema dirá que es un Pájaro, y acertará.

Si el programa hace todo lo anterior, funciona bastante bien y usted puede ensayar con algo un poco más audaz.

Pruebe con dos nodos. Utilice el primero para establecer si en lo que está usted pensando es una máquina o un animal. Por ejemplo:

Variables	*Resultados*
Plumas	Animal
Metal	Máquina

En el segundo nodo se establecerá si el objeto es un pájaro o un avión. Por ejemplo:

Variables	*Resultados*
Plumas	Pájaro
Metal	Avión
Animal	
Máquina	

Cuando introduzca usted ejemplos en el nodo 1, piense primero en un animal y responda a las preguntas de acuerdo con ello. Luego piense en una máquina y responda adecuadamente. Basta con dos ejemplos.

Introduzca a continuación dos ejemplos en el nodo 2. Uno pensando en un pájaro y otro en un avión.

Cuide especialmente de deletrear las palabras del mismo modo cada vez que las teclea, pues de otro modo no va a producirse correctamente el emparejamiento de las cadenas de caracteres.

Pase luego a ejercitar al experto, y, cuando éste termine, póngalo en funcionamiento normal.

Lo primero que pregunta es sobre Plumas. Suponga que piensa usted en un Pájaro, y responde 1. El sistema entonces sugiere Animal como lo más probable, y pregunta, con el nodo 2, sobre Metal. Responda 0, y el sistema dirá que es un Pájaro.

Lo que ha sucedido es que una sola variable, Plumas, ha sido suficiente para resolver el nodo 1 y permitir al experto deducir que el objeto era un animal. Ahora bien, si miramos en el nodo 2, vemos que en él también están las Plumas (cuyo valor ya está dado), y que el sistema ya había deducido unos valores para Animal y Máquina (1 y 0, respectivamente), de modo que lo único sobre lo que no está aún seguro es sobre Metal, así que pregunta por ello, obtiene una respuesta y ya puede resolver correctamente el nodo 2, diciendo que es un Pájaro.

Haga otra prueba, pensando en un Avión.

Pregunta: ¿Plumas?. Respuesta: 0. El sistema deduce que es una Máquina, y de nuevo pregunta por Metal. Responda 1, y el sistema deducirá que es un Avión.

Afortunadamente, sus doloridos dedos ya estarán recuperándose y usted se encontrará algo mejor después de todo aquel esfuerzo.

Y la pregunta es: ¿puede el sistema hacer algo útil?. Algo que no sea un juego aparentemente trivial. Bien, pues podemos hacer que pruebe con un problema de la vida real, como, por ejemplo, tratar de diagnósticar fallos en una grabadora de cassettes.

Supongamos que los fallos que se pueden observar en la grabadora son los siguientes:

1. No se enciende la luz
2. La cinta no se mueve
3. El aparato no graba
4. Sonido intermitente
5. Sonido distorsionado
6. Velocidad variable
7. Zumbido

Y que estos fallos pueden proceder de cualquier combinación de las siguientes causas:

1. Aparato no conectado
2. Botón de «Pausa» pulsado
3. Cinta atascada
4. Cinta mal insertada
5. Lengüeta de la cassette quitada
6. Cabeza sucia
7. Cinta deformada
8. Mala grabación
9. Problema de amplificador
10. Cabestrante sucio
11. Cables mal conectados

Y que la acción a ejecutar para resolver el problema fuera una de las siguientes:

1. Conectar el aparato
2. Apretar «Pausa»
3. Cambiar cassette
4. Reinsertar cassette
5. Limpiar cabezas
6. Volver a grabar cinta
7. Comprobar amplificador
8. Limpiar cabestrante
9. Comprobar cables

Por tanto, inicialice un sistema con dos nodos. El primero de ellos servirá para encontrar qué es lo que falla, de modo que tendrá como variables los siete fallos y como resultados las 11 causas.

El segundo nodo servirá para sugerir la acción de reparación, por lo que tendrá las 11 causas como variables y las 9 reparaciones como resultados.

Introduzca ejemplos. Para el primer nodo, dé 11 ejemplos, uno para cada causa, y responda a las preguntas del sistema pensando en cada causa específica. Para el segundo, dé 9 ejemplos, uno para cada acción de reparación, y responda a las preguntas sobre las causas dando un «1» si la causa puede quedar reparada mediante la acción, y un «0» en caso contrario.

Luego ejercite al experto (o también, en lugar de introducir ejemplos y hacer que se ejercite posteriormente, podría haber llevado a cabo una sesión de entrenamiento). Hay que señalar que, conforme el sistema se

va haciendo más grande, las sesiones de entrenamiento o de ejercicio tardan más; no piense, pues, que el ordenador se ha estropeado si tarda algún tiempo en resolver el problema.

Cuando el ordenador termine, habrá calculado un conjunto de reglas, y puede usted ponerlo en el modo de funcionamiento normal, en el que el sistema va a tratar de decidir qué es lo que falla en su grabadora de cassettes.

Los detalles exactos del funcionamiento dependerán de los ejemplos que haya dado usted, pero, a título ilustrativo, he aquí lo que ocurrió en una sesión:

El sistema preguntó si la luz no se encendía. Se le respondió «1» (sí, no se enciende), con lo que sugirió «Aparato no conectado» como lo más probable, e inmediatamente pasó al nodo 2, con el que aconsejó «Conectar el aparato».

En la siguiente sesión decidimos que el problema era Sonido Intermitente.

Preguntó sobre la luz. Respuesta: «0».
Preguntó sobre velocidad errática. Respuesta:«0».
Preguntó si la cinta no se movía. Respuesta: «0».
Preguntó sobre sonido intermitente. Respuesta: «1».
Preguntó si había zumbido. Respuesta: «0».
Preguntó si no registraba. Respuesta: «0».

A partir de ahí, el sistema dedujo que la causa más probable era «Mala Grabación», y, pasando al nodo 2, aconsejó «Limpiar Cabestrante», sin plantear más preguntas.

¿Le parece razonable?. Bueno, quizás no del todo: idealmente, uno podría esperar que le hubiese aconsejado «Limpiar Cabezas». Lo más probable es que la colección de ejemplos que se le dió no fuera muy completa. Pero la línea de razonamiento no es tan mala: al fin y al cabo, un cabestrante sucio puede originar una mala grabación, lo cual puede causar sonido intermitente. Parece que todo se puede arreglar con un poco más de tiempo de entrenamiento (a menos, desde luego, que limpiando el cabestrante hayan desaparecido todos sus problemas).

Pruebe ahora con «La cinta no se mueve». El sistema hace las mismas tres preguntas de antes. Cuando usted responde con 1 a «La cinta no se mueve», él pregunta sobre «Zumbido»; tras responder 0, sugiere «Cinta mal insertada», y, pasando al nodo 2, le aconseja que reinserte la cassette. No está nada mal.

Pruebe usted mismo: éstos son los ejemplos con los que se entrenó al sistema:

Nodo 1 (Diagnosis)

Variables (Fallos)	Resultados (Causas) 1	2	3	4	5	6	7	8	9	10	11
1	1	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	0	0	0	0	0	0	0
3	1	1	1	1	1	0	0	0	0	0	0
4	0	0	0	0	0	1	1	1	0	0	0
5	0	0	0	0	0	1	1	1	1	1	1
6	0	0	0	0	0	0	1	0	0	1	0
7	0	0	0	0	0	0	0	0	1	0	1

Nodo 2 (Acción de reparación)

Variables (Causas)	Resultados (Reparaciones) 1	2	3	4	5	6	7	8	9
1	1	0	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0
5	0	0	1	0	0	0	0	0	0
6	0	0	0	0	1	0	0	0	0
7	0	0	1	0	0	0	0	0	0
8	0	0	0	0	0	1	0	0	0
9	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	1	0
11	0	0	0	0	0	0	0	0	1

Una vez que lo haya escrito y ejecutado, enseguida se dará usted cuenta de que con este sistema puede realmente hacer algo útil. Después de todo, el programa, tal como está, puede tener muchos más nodos, muchas más variables y un conjunto de interconexiones mucho más variado.

Quizás le parezca que, en muchos de los ejemplos dados hasta ahora, sería más natural responder Sí o No, en lugar de 1 y 0, y probablemente tenga usted razón. De hecho, se puede adaptar el programa en muchos sentidos, para que, dada una aplicación, se comporte de un modo más natural. Sólo un breve recordatorio (de nuevo): las variables de entrada no tienen por qué valer siempre cero o uno. Podrían ser números reales, como las cifras de precipitación, por ejemplo. Recuerde este

detalle, que puede ayudarle a mejorar el funcionamiento del sistema cuando esté usted desarrollando una idea.

Observe que en el Apple II y en muchos otros micros todas las variables y las matrices se inicializan a cero cuando se teclea RUN. Si su micro no lo hace, entonces tiene usted que poner a cero, al comienzo de cada ejecución, todas las matrices y algunas variables (puede hacerlo al comienzo del programa). Asimismo, a algunos micros (incluyendo el Spectrum) hay que decirles inicialmente qué variables se suponen nulas, con sentencias como

```
LET C=0
```

Algunos micros suponen que C es cero si no se ha mencionado previamente, pero en el Spectrum hay que ponerla a cero explícitamente.

8. CÓMO UTILIZAR SU EXPERTO

8.1 LA ELECCIÓN DE UN PROBLEMA

Supongamos que ahora desea usted construir su propio sistema experto. No el descrito anteriormente en este libro, sino algo que sea exclusivamente suyo. Bien, pues lo primero que tiene que decidir es: ¿Sobre qué ha de ser experto el sistema?

Si quiere, usted puede responder que ha de ser experto en absolutamente todo lo del Universo Entero Conocido. Nada hay de malo en tal aspiración. Pero si procede usted así, probablemente va a terminar con algo parecido al sistema descrito en el capítulo anterior. Al fin y al cabo, para que sea experto en todo, se necesita un diseño muy general, con el mínimo de concepciones previas. El inconveniente de este enfoque es que, si bien el sistema podría funcionar para muchos problemas diferentes, no sería extraordinariamente bueno con ninguno de ellos.

Lo mejor es que elija usted un área de experiencia que no sea demasiado amplia ni demasiado reducida. Afirmación ésta que puede parecer poco precisa (y ciertamente lo es), pero hay algunas razones detrás de ella.

Supongamos que escoge usted un campo demasiado reducido. Como, por ejemplo, el diagnóstico de averías en un motor de automóvil. Podría usted pensar que este campo es muy interesante: después de todo, los mecánicos cobran bastante, y conseguir que un ordenador haga su trabajo puede ser práctico. Pero consideremos más detenidamente el problema.

Pongamos que su coche no arranca por la mañana. Lo que quiere usted es que el ordenador diagnostique lo que falla. Por tanto, lea el manual del coche, y probablemente encontrará que la causa de que no arranque es una de las siguientes: falta de gasolina, batería descargada,

agua en el distribuidor, o bujías sucias. Introduzca eso en un sistema experto, y ejecute (mentalmente) el programa. Y sobre la pantalla aparecería la pregunta: ¿hay gasolina en el depósito? Y usted iría corriendo al coche para echar un vistazo y poder dar la respuesta. Si ésta fuera positiva, el sistema le preguntaría: ¿hay agua en el distribuidor? Y de nuevo, a correr hacia el coche a mirar. Y si no la hubiera, cinco minutos después estaría usted desenroscando una bujía para ver si está sucia...

Para hacer eso no es necesario enchufar el ordenador. No tiene más que leer el manual, y ahorrará electricidad.

La explicación es que el diagnóstico de por qué un automóvil no arranca es trivial. El verdadero esfuerzo radica en rebuscar en el coche para encontrar la información necesaria.

En este caso, el ordenador no puede relevarle de la tarea de husmear en el coche. Por tanto, lo que realmente necesita usted es un mecánico que le ayude.

Desde luego, puede haber ciertas tareas en mecánica de automoviles que podrían beneficiarse de un sistema experto. Imagine que dirige usted un taller en el que tiene a unos mecánicos relativamente inexpertos comprobando diferentes vehículos y realizando ciertos trabajos complicados sobre ellos. Podría entonces ser útil el disponer de una pantalla que les aconseje cómo han de proceder. También podría usted darles un manual de reparaciones. Pero, si sólo se trata de un coche que no arranca, ciertamente es un problema demasiado reducido para que valga la pena abordarlo. El problema debe ser suficientemente «amplio» para que el ordenador haga algo realmente útil. Y será mucho más útil si la tarea no está ya resuelta por algún otro medio (tal como un manual de reparaciones).

En el otro extremo de la escala, el problema no debe ser demasiado amplio. Las razones para ello son también de tipo práctico. En primer lugar, el problema práctico de llegar a construir un sistema experto con suficiente experiencia en un área muy extensa. Conforme la extensión aumenta, también aumenta el esfuerzo necesario de su parte para conseguir una realización completa. Y si la realización no es completa, la utilidad del sistema es más que dudosa.

Los Sistemas Expertos de la Vida Real no son expertos en, por ejemplo, todo el campo del diagnóstico médico, sino en un subcampo reducido, en el que funcionan muy bien. Un sistema que tratase de diagnosticar todo, y que frecuentemente haría diagnósticos falsos, propendería a matar tantos pacientes como curase.

Finalmente, las posibilidades de que el sistema sea útil serán máxi-

mas si se encuentra algún método para que funcione. Esto parece una perogrullada, pero imagine que tiene usted un sistema al que trata de hacer experto en acertar quinielas de fútbol. Ahora bien, aunque sea preliminarmente, ¿cómo podría usted llegar a generar tal sistema?. Algunos deportes y juegos de azar pueden ser susceptibles de predicciones por ordenador. Pero, ¿el fútbol?. Basta con que algunos jugadores desfallezcan en el campo como consecuencia de una resaca, y el resultado del juego será totalmente diferente de cualquier resultado que el ordenador hubiera podido imaginar. El problema es demasiado difuso para que ningún ordenador pueda manejarlo. Las reglas que gobiernan el juego real no están en absoluto claras, ni se ve que haya ningún método con el que puedan descubrirse tales reglas.

En general, puede usted hacerse una idea de hasta qué punto un tema es adecuado para un sistema experto (o para cualquier programa) preguntándose a sí mismo si hay muchas cosas que puedan medirse.

Si las hay, entonces tiene posibilidades. Y si no, olvídese.

Desde luego, por «medir» no queremos significar, necesariamente, longitud, anchura y profundidad. Una respuesta sí/no corresponde al tipo de medida que hace feliz a un ordenador. Pero, globalmente, tiene usted que poder describir el tema en función de un conjunto de medidas de algún tipo. Si la situación contiene algo que de ningún modo puede reducirse a una cantidad medible (como el toque de balón del jugador X), entonces es poco probable que pueda llevarse a un ordenador.

8.2 EL ANÁLISIS DEL PROBLEMA

Una vez que se le ha ocurrido un área de problemas que parece valer la pena, el siguiente paso es empezar a analizarla.

Ante todo, se necesita tener una perspectiva general, cosa que, normalmente, ya tendrá usted. Al fin y al cabo, la mayoría de los sistemas no se escriben accidentalmente: surgen porque un especialista en algún campo piensa que un ordenador podría ayudarle, o porque un informático conoce a un especialista que tiene un problema. Si no tiene usted esa visión general, es el momento de introducirse poco a poco en el asunto, principalmente para ver si estaba usted en lo cierto al pensar que se podría resolver con un ordenador, es decir, comprobar que, efectivamente, es un problema adecuado.

Luego, el siguiente paso típico consiste en abordar a un experto y rebuscar en su mente. Ello puede hacerse siguiendo una secuencia lo más

ordenada posible, que consiste en averiguar los resultados, la evidencia medible (las variables) y el razonamiento que engarza unos con otras.

Los resultados pueden ser muy simples: oro en aquellas colinas, bronquitis en aquestos pulmones, o cosas así. Ahora bien, ¿lo importante es la presencia o la ausencia de un resultado, o hay alguna otra medida asociada con ese resultado?. Como una probabilidad, por ejemplo. Es muy importante sentarse y pensar qué es lo que se quiere obtener del sistema y en qué forma. Una lista de posibles diagnósticos, o conclusiones, o recomendaciones, y cómo han de medirse esos resultados.

Las variables son los elementos de evidencia que el experto humano tiene a su disposición. Tendrá usted que llegar a saber cuáles son todas esas variables y cómo se miden. Una vez que le ha preguntado al experto qué es lo que espera encontrar, tiene que saber qué es lo que tiene en cuenta para obtener sus conclusiones.

Los encadenamientos entre ambas cosas son las reglas que aplica el experto. ¿Con qué «programa interno» trabaja el experto para llegar a sus conclusiones?. Y esto puede que sea lo más difícil. Es muy posible que el experto humano no conozca exactamente las reglas que utiliza. Por tanto, todo lo que usted puede hacer es un primer intento para averiguar cómo cree él que procede, y programarlo para ver qué pasa.

Una vez que haya acumulado toda esta información inicial, puede ser que vea una forma natural de construir el programa.

Por ejemplo, con nuestro sistema de propósito absolutamente general, el requisito de generalidad dictaba el que hubiera de funcionar mediante autoaprendizaje, elaborando sus reglas a partir de ejemplos. También resultaba evidente que la estructura concreta que se usara para las inferencias, no era demasiado crítica, porque la que quizas fuese buena para una aplicación podría no serlo tanto para otra (y, por tanto, sea cual sea la situación, la utilidad del sistema que hemos construido siempre será pequeña...).

Pero si tiene usted ante sí una lista concreta de resultados, una lista concreta de variables y una lista concreta de reglas, entonces la situación es diferente. Si los tipos de las reglas son muy diferentes entre sí, puede que tenga usted que escribir instrucciones específicas para cada una de ellas. Si pueden reducirse a un formato común, puede almacenarlas en un fichero como si fuesen datos y escribir una rutina que trabaje sobre ellas. Si todas ellas se basan en conectivas lógicas, puede escribir fácilmente un programa deductivo sencillo que le permita llegar a conclusiones seguras. Si tienen asociados elementos probabilísticos, necesitará algún método para tener en cuenta las probabilidades.

Indicar que no se puede aconsejar con detalle qué es lo que hay que hacer, y que usted debe averiguarlo por sí mismo, puede parecer que es como no decir nada. Pero, de hecho, eso es realmente lo que tiene usted que hacer. Puede tener en cuenta algunas de las ideas de este libro, que quizás le ayudarán a establecer un enfoque. Pero de igual modo que un sistema experto de propósito general no funciona bien con los detalles muy concretos, tampoco funcionaría bien un método general para construir sistemas expertos.

Las personas que han construido por su cuenta sistemas expertos, suelen informar de que la etapa más laboriosa es la de acumular inicialmente los resultados, las variables y las reglas. Una vez que consiguieron extraer esta información de un experto humano y escribirlas en un papel, el resto comenzó a aparecer claramente ante ellos. Pero, aunque en esta etapa no aparezca con toda claridad cuál es la estructura a utilizar, hay un hecho del que también se informa que facilita las cosas. Y este hecho es que el proceso no termina aquí.

Con la información inicial, se elabora algo que funcione en el ordenador, una especie de programa tentativo que se piense que podría servir. Entonces, se ejecuta con algunos ejemplos y se pasan los resultados al experto humano para que los comente. Normalmente, el sistema cometerá errores, y se establece un proceso de realimentación entre el informático, el experto humano y el ordenador, durante el cual se van alterando progresivamente las reglas del programa hasta que éste empieza a funcionar razonablemente bien.

Puede que piense usted que, en cierto modo, este proceso es poco sistemático. Después de todo, si usted se pusiera a escribir un programa de nóminas sin tener mucha idea de cómo hay que calcular las nóminas, la gente podría pensar que se ha equivocado usted de oficio y que más le valdría dedicarse a la cría de gallinas, por ejemplo.

Pero así es como parece que se han desarrollado los sistemas expertos. Y puede argumentarse que es un método razonable en una situación en la que nadie sabe realmente cómo hay que hacer las cosas. Después de todo, si llega a funcionar, seguramente el fin justifique los medios utilizados.

Y hay que admitir que, de hecho, hay quien ha escrito incluso programas de nóminas, siguiendo un método no muy diferente de éste, y se las ha ingeniado para vender el producto.

En realidad, lo más difícil en el análisis del problema es el responder a la pregunta de por qué ha de haber un experto humano decidido a gastar su tiempo, en principio valioso, en transmitir los secretos de su arte a

alguien que va a ponerlos en un disco, hacer un millón de copias y venderlas a 1.000 pts. cada una. Tal experto probablemente estaría loco. Una de dos: o bien realmente no le va a transmitir su experiencia al sistema, con lo cual éste será completamente inútil, y por tanto nada va a perjudicar a sus intereses, o bien no le importa ver cómo se devalúa su experiencia y se queda sin negocio. A fin de cuentas, ¿dónde han ido a parar los que hacían las nóminas a mano?. ¿No estarán en la cola del paro, esperando encontrarse una vez más con aquél encantador informático?

Quizás debería usted ser generoso y ofrecer a su experto humano una parte de los derechos que cobre por las copias de su sistema. Quizás debería usted ser generoso y ofrecerme también a mí una parte.

9. SISTEMAS EXPERTOS DE GRAN ENVERGADURA

9.1 EL MYCIN: DIAGNÓSTICOS MÉDICOS

Hasta ahora, todo lo que hemos considerado es «el sistema experto»: el nuestro, de propósito completamente general y de fabricación casera. Pero, naturalmente, hay otros. ¿En qué medida se parece el nuestro a ellos? Pues bien, la mejor respuesta consiste en describir algunos otros y ver dónde radican los parecidos, si los hay.

MYCIN es un sistema experto diseñado para realizar diagnósticos médicos. Concretamente, se diseñó para trabajar en el área de septicemia y meningitis, de modo que, a partir de la evidencia que se le presenta, hace un diagnóstico adecuado y recomienda un tratamiento con drogas para la enfermedad infecciosa diagnosticada. Consta de un total de 450 reglas desarrolladas con la ayuda del Grupo de Enfermedades Infecciosas de Stanford.

Su característica más importante (y la que puede dar lugar a la mayoría de las dificultades) radica en su uso de las probabilidades. El diagnóstico médico es una ciencia inexacta. Si un paciente presenta un determinado conjunto de síntomas, ello podría indicar cierta enfermedad, pero la correspondencia no es casi nunca absoluta. Consideremos la diferencia entre un sistema de diagnóstico médico y un sistema experto en el campo de la química, por ejemplo. Para facilitar las cosas, consideraremos sistemas hipotéticos, lo que, por otra parte, nos evitará la intrusión en el coto de los expertos humanos.

Supongamos que tenemos un sistema experto en análisis químicos y que uno de los elementos de información que le damos es el resultado de un experimento con papel de tornasol. Con cierta disolución, el papel se

pone, por ejemplo, rojo. Con ello, el experto puede «diagnosticar» fácilmente que la disolución es ácida. No hay duda alguna.

Pero pasemos al sistema de diagnóstico médico, e informémosle de que el paciente bajo consideración tiene una tos horrible. Pues bien, ello podría indicar que estamos ante un caso de bronquitis, o de tuberculosis, o... simplemente, ante una tos horrible. No hay una certidumbre absoluta sobre el significado de la evidencia.

Y, de un modo u otro, el sistema experto debe ser capaz de hacer frente a esta incertidumbre. Nuestro propio sistema experto lo hacía, en cierta medida, pero de un modo no muy preciso. Y si se acuerda usted de la discusión anterior sobre las probabilidades, recordará cuán difícil es tratar con un problema como éste.

La forma adoptada en MYCIN para atacar el problema consiste en asignar un Factor de Certidumbre a cada una de sus 450 reglas. Podemos, así, considerar que MYCIN contiene un conjunto de reglas de la forma SI...ENTONCES... (IF...THEN...) con certidumbre P.

Y observe que hemos utilizado la expresión «Factor de Certidumbre» en lugar de la palabra «Probabilidad». Y la pregunta es: ¿por qué?; ¿son acaso diferentes?

Pues bien, si ha leido usted todos los capítulos anteriores de este libro, ya lo sabe todo sobre las probabilidades, pero de lo que quizás no se dé cuenta es de que existe más de un tipo de probabilidad. El tipo que hemos considerado hasta aquí (y el que seguiremos considerando) es el de las probabilidades estadísticas. Toda la teoría de la probabilidad estadística se basa en la hipótesis de que, contando con suficientes ejemplos, las estadísticas describirían con precisión el comportamiento del sistema que estamos considerando. Es el enfoque frecuencial de la probabilidad.

Sin embargo, algunos consideran que éste no es un enfoque acertado para utilizar con sistemas de inferencias (sistemas que modifican su grado de creencia en un resultado, dependiendo de las entradas que reciben). Para estos sistemas, dicen, es mejor una teoría de la Probabilidad Lógica que una teoría de la Probabilidad Estadística, porque en el caso de un sistema de inferencias no hay realmente un modelo frecuencial de lo que está sucediendo. Entrar en el detalle de la teoría de la Probabilidad Lógica desbordaría el marco de este libro, pero, como pequeña compensación, este autor sugiere que, si se acoge usted a un modelo estadístico, no va a equivocarse demasiado, y que los teóricos de la Probabilidad Lógica no le van a concretar muy bien qué cálculos tiene usted que hacer en el caso de que acepte sus teorías. (Esta crítica de la Proba-

bilidad Lógica se conoce como la Teoría de la Zorra y las Uvas Verdes). En otras palabras, aunque usted conociese la susodicha teoría, ello seguramente no le serviría para mejorar su programa.

En cualquier caso, la Probabilidad Lógica es el enfoque que utiliza MYCIN, y el efecto práctico es que los Factores de Certidumbre de MYCIN vienen a ser, aproximadamente, lo que para la mayoría de las personas serían probabilidades condicionales de la forma P(H:E) (probabilidad de cierta hipótesis, dada cierta evidencia). Como realmente no son probabilidades en el sentido en que las hemos estudiado, no se aplican los cálculos que hemos visto anteriormente; MYCIN utiliza un método completamente *ad hoc* para ir evaluando sus factores de certidumbre conforme avanza por el programa.

Pero, para empezar por el principio, ¿de dónde salen esos factores de certidumbre?. En el caso de MYCIN, proceden de los expertos humanos que proporcionaron inicialmente las reglas. Cada vez que sugerían una regla, asignaban su grado de confianza en esa regla en una escala de 1 a 10.

Y ahora surge la observación que antes hicimos en relación con las probabilidades: ¿cómo podemos saber si esas probabilidades son correctas? Pues bien, en realidad no podemos saberlo. Seguramente son más o menos correctas (y el hecho de que MYCIN dé buenos resutados apoya la sugerencia de que son bastante buenas), pero el método es básicamente *ad hoc* (lo cual significa que no dejaría satisfecho a ningún estadístico).

Una vez establecidas las reglas con sus certidumbres asociadas, MYCIN funciona mediante encadenamiento hacia atrás, partiendo de un posible resultado y viendo si se puede creer o no en ese resultado. Cuando ha recogido todos los elementos que necesita para un resultado determinado, establece un juicio sobre ese resultado, calculado en base a los factores de certidumbre asociados con todas las reglas que se han utilizado para llegar a él.

Por ejemplo, si el resultado fuese Z, podría ocurrir que hubiese sido necesario establecer X e Y para deducir Z. Y las reglas utilizadas para establecer X e Y podrían tener asociadas las certidumbres P y Q. Si tanto P como Q tuviesen el valor máximo (10), entonces se deduciría Z con seguridad. Pero, si son menores (y generalmente lo son), entonces no se infiere necesariamente Z, sino sólo con cierta cantidad de certidumbre.

Y, si recordamos nuestra discusión anterior sobre probabilidades, no es tarea fácil calcular exactamente cuál será, en estas circunstancias,

la certidumbre de Z. Ello dependerá en gran medida de la forma exacta de todos los elementos involucrados y de cómo se interrelacionan.

Por tanto, MYCIN, en lugar de tratar de obtener una solución exacta a este problema, acumula simplemente todas las certidumbres involucradas para llegar a una idea aproximada de la magnitud relativa de las respuestas.

Posiblemente, la observación más importante sea que lo que hace MYCIN no es proponer un diagnóstico y dar la certidumbre exacta en que ese diagnóstico sea cierto, sino llegar a todo un conjunto de diagnósticos, cada uno de ellos con una especie de «puntuación» de certidumbre asociada. Por encima de un cierto valor (arbitrario), todos esos diagnósticos se aceptan como probables en cierta medida, y lo que se le presenta al usuario es una lista de posibilidades.

Matemáticamente, el procedimiento es poco sólido, pero, sin embargo, la evidencia es que en la práctica funciona muy bien. Tanto peor, desde luego, para las matemáticas...

De todas formas, el diagnóstico médico es en sí mismo un procedimiento algo difuso. Un médico no establece una probabilidad exacta sobre cada uno de sus pacientes; estima, simplemente, que cierto diagnóstico parece bastante verosímil, existiendo, quizás, otros diagnósticos como posibilidades adicionales. Es más, podría ser que un paciente no sufriese sólo de una cosa, sino de varias simultáneamente, y, en este caso, establecer exactamente un solo diagnóstico sería excluir la posibilidad, perfectamente válida, de que no haya sólo uno, sino varios.

Se ha informado de que los profesionales médicos que han trabajado con MYCIN se han sentido bastante satisfechos con él, y han estimado que sus habilidades son tan buenas como las suyas propias. Y posiblemente ésta sea la mejor prueba de que el invento funciona.

El siguiente aspecto a observar de MYCIN es su uso del lenguaje inglés. Cuando precisa que el usuario le dé alguna información, se la pide en inglés. Y el usuario puede introducir la información de una forma bastante natural y próxima al inglés. Esto parece cómodo, pero concretemos algo más.

El asunto tiene dos facetas: la que concierne a la aceptación por parte del usuario del producto final (¿ha valido la pena hacerlo?) y la relativa a la realización propiamente dicha (¿ha sido fácil hacerlo?).

Empezando por la primera faceta, se considera que una de las ventajas de los sistemas expertos es que cualquiera puede utilizarlos con un

mínimo de conocimientos previos sobre ordenadores. Ciertamente, es fácil darse cuenta de que los usuarios responderán más favorablemente a un sistema que utilice su lenguaje que a otro que les obligue a hablar en BASIC, por ejemplo. Así, a los médicos que utilizaron MYCIN el lenguaje les pareció fácil, y seguramente no habrían perdido el tiempo evaluando un sistema de difícil uso. Y si nadie quisiera perder el tiempo utilizando un sistema, podría concluirse que su utilidad a largo plazo sería practicamente nula.

¿Y en cuanto a la realización?. Es bien sabido que el procesamiento del lenguaje natural es una de las tareas más difíciles que puede haber. ¿Cómo abordar el problema?

La respuesta tiene algo de trampa, porque MYCIN, de hecho, no hace en absoluto un procesamiento completo del lenguaje natural.

El truco radica en que en toda profesión existe la tendencia a jugar al juego de su propio y reducido lenguaje, con palabras especiales y formas estereotipadas de decir las cosas, propias de cada profesión. Hay muchas razones, y algunas buenas, para que así sea. A un nivel algo cínico, podría señalarse el hecho de que hasta los niños pequeños tienen a menudo su propio lenguaje privado, que utilizan para excluir a los extraños de sus conversaciones, y que sirve para aumentar la solidaridad con sus amigos. Y los adultos también tienen esos lenguajes privados (incluyendo a los informáticos, que son los peores de todos). A un nivel más serio, cuando uno habla sobre conceptos precisos tiene que utilizar las palabras de un modo preciso e invariante. Ello conduce a un lenguaje especializado sobre el tema, que podría parecer lenguaje natural a una persona ajena al mismo, pero que en realidad es muy diferente.

Tomemos, por ejemplo, las palabras «crónico» y «agudo». En la jerga médica, esas palabras se refieren simplemente a la duración de una enfermedad: a si está presente durante mucho (crónica) o poco tiempo (aguda). Así, cuando un médico dice que su paciente tiene una tos crónica, no quiere decir que esa tos sea digna de publicarse en la prensa. Ni cuando dice que la tos es aguda, está pensando que el paciente vaya a caer fulminado de un momento a otro. Simplemente, está haciendo una observación sobre el tiempo que el paciente lleva tosiendo.

Es posible que todo esto parezca una divagación innecesaria, pero puede ayudar mucho para la realización de un sistema experto. En el caso de MYCIN, se descubrió que los médicos que trabajan en este campo del diagnóstico utilizan las palabras de un modo muy preciso, y hacen comentarios muy estereotipados, mucho más de lo que puede hacer la gente en una conversación normal. Y la ventaja de este hecho es que se pudo definir fácilmente un subconjunto muy limitado del inglés con el

que fue posible expresar sin grandes dificultades todo cuanto puede decirse sobre el tema. Se adaptaron al programa las frases y las formas gramaticales estandarizadas, y el resultado fue un pequeño subconjunto del inglés fácil de programar.

A los médicos les satisfizo el resultado, porque ellos también hablan, posiblemente sin percatarse de ello, en un subconjunto muy reducido del lenguaje natural, por lo menos cuando discuten sobre su trabajo. Por lo que uno sabe, sólo se transforman en magníficos oradores cuando vuelven a su casa por la tarde, pero esto ya es otro cantar.

Todo esto tiene cierta relación con las observaciones que haremos luego acerca de DENDRAL, que, como veremos, no utiliza en absoluto el inglés. Lo que utiliza es un lenguaje gráfico adecuado para las actividades particulares de los químicos, que puede considerarse como un subconjunto muy reducido del lenguaje natural. La observación a hacer es la misma de antes: Si se modela el sistema utilizando el lenguaje de los expertos humanos que van a utilizarlo, se aumentarán las posibilidades de que los usuarios lo acepten, y se podrá sacar partido más fácilmente del conocimiento que ya existe en ese campo, que, en alguna medida, habrá dado forma al lenguaje que se utiliza para describirlo.

Sin embargo, lo malo (siempre hay algo malo) es que, si se procede así, se tendrá un sistema difícil de adaptar a otras áreas de experiencia, debido simplemente a las diferencias existentes entre los lenguajes que se utilizan para describir la experiencia en campos distintos.

En el tema de la aceptación por el usuario, MYCIN tiene una capacidad que también poseen otros sistemas expertos, y que se comenta con frecuencia: su habilidad para explicar al usuario por qué hace lo que hace.

Por utilizar un ejemplo sencillo, imagine que tiene usted un sistema experto, y que éste le pregunta si tiene usted tos. En lugar de responder simplemente sí o no, podría usted preguntarle al experto: ¿por qué?. Es decir, ¿por qué me planteas esa pregunta?. El sistema podría entonces escribir un mensaje diciendo que la tos indica a veces alguna enfermedad pulmonar. O algún otro comentario de ese tipo.

Ahora bien, los programadores se darán cuenta de que, en su forma más sencilla, esto puede ser simplemente un comentario del programa. Todo lo que hay que hacer es escribir, siempre que se programa una regla, un pequeño texto que explique el objetivo de esa regla. De este modo, cuando el sistema plantee una pregunta que involucre a esa regla, puede hacer uso, si el usuario lo pide, de ese texto a modo de explicación. Recuerda algo a las sentencias REM de BASIC.

Dicho de otro modo, el asunto no parece que requiera mucho ingenio. Entonces, ¿por qué tanto interés en él?

El motivo de incluir tal cosa es conseguir la aceptación, por parte del usuario, del producto final. A los usuarios, especialmente a aquellos que no saben mucho sobre ordenadores, les causa buena impresión un sistema que actúa de una forma tan humana que puede explicarse cuando se le pregunta. Y, como señalábamos más arriba, no tiene mucho sentido escribir un sistema que nadie está dispuesto a utilizar.

Pero hay algo más detrás de todo ello. Considere al programador (usted, por ejemplo, cuando escribe el sistema). Las REM del programa están muy bien. Sin duda, ayudan al diseño y a la depuración, porque le recuerdan por qué ha incluido usted cada segmento de programa, y qué se supone que hace ese segmento. Pero la pega con las REM es que hay que listar todo el programa para poder leerlas. Puede parecer una queja trivial, pero, si el programa es larguísimo, inacabable, tendría que haber un método mejor para tener registradas las explicaciones.

Y eso es, en cierta medida, lo que representan los «¿por qués?» de los sistemas expertos. Son como REM «vivas» que pueden llamarse en cualquier momento para que le recuerden a uno lo que pasa, sin necesidad de interrumpir nada.

Alejémonos ahora un poco más del programa real y recordemos los problemas que suelen encontrarse al construir un sistema experto: los relativos a la ingeniería y la adquisición del conocimiento. El programa contiene un conjunto de reglas y de explicaciones sobre ellas, y, normalmente, el programa todavía no funciona a la perfección. El experto humano se encuentra sentado ante la pantalla trabajando con un ejemplo. De pronto, la máquina plantea una pregunta estúpida. En ese momento, el experto (humano) puede preguntar «¿por qué?» y así obtener alguna idea de qué es lo que falla, con miras a corregir el funcionamiento mediante la inclusión de alguna otra regla o la modificación de las reglas ya existentes.

Todo eso no sería muy fácil de hacer si el sistema se limitase, realmente, a imprimir un comentario estándar. No cabe duda de que algo un poco más sofisticado ayudaría a esa tarea.

En el caso de MYCIN, este «algo-más-sofisticado» es TEIRESIAS, un sistema que permite modificar las reglas y explicar las acciones de MYCIN. Esencialmente, se parece mucho a una facilidad de trazado (*trace*) y volcado (*dump*), con la gran ventaja de que es algo más cercano al usuario que la técnica tradicional consistente en llenar una caja de pa-

pel de impresora con el contenido en hexadecimal de la memoria central.

El experto humano sentado ante la pantalla de MYCIN puede preguntar «¿por qué?» como respuesta a una solicitud de información, y obtener un resumen de la línea de razonamiento seguida hasta ese punto. Concretamente, el sistema puede escribir la regla que ha provocado la petición de información, y mostrar el valor de todas las demás entradas (si las hay) de esa regla.

Recuerde usted el sistema de la primera parte de este libro, e imagine que le ha planteado el problema de identificar a un objeto como un pájaro, un avión o un planeador. En algún momento, el sistema le pregunta si el objeto tiene motor, y usted a su vez le pregunta: «¿por qué?».

Evidentemente, sería muy fácil comprobar a través del nodo actual cuál es el estado de las otras variables. Y en ese caso, el sistema podría indicarle que ya había determinado que el objeto tiene alas y que no tiene pico, y (añadiendo algunas sentencias en el programa) señalar que, si obtuviese una respuesta «Sí» a motor, deduciría que el objeto es un avión, mientras que con la respuesta «No» deduciría que es un planeador.

Y si las reglas con las que estaba trabajando no eran buenas, quizás anunciaría su intención de deducir Planeador en el caso de que el objeto tuviese un motor, lo cual sería incorrecto, y permitiría al usuario saber que esta regla concreta necesita modificación.

Naturalmente, un sistema como MYCIN necesita un programa mucho más largo para poder adaptar sus sentencias, pero el principio es el mismo. El «*¿por qué?*» proporciona una instantánea sobre la línea de razonamiento del sistema en cada momento, lo que constituye una ayuda útil en el desarrollo inicial y en la depuración, y también sirve para tranquilizar a los eventuales usuarios de que el sistema no está funcionando al azar.

Otra facilidad de TEIRESIAS es la pregunta «¿cómo?», que puede aplicarse a cualquier afirmación. Pensando de nuevo en nuestro ejemplo del pájaro/avión/planeador, podríamos haber incluido un trozo de programa que permitiese al usuario preguntar: «*¿cómo alas?*». Es decir, el sistema cree que el objeto en cuestión tiene alas; ¿cómo ha llegado a creer tal cosa?. En este ejemplo, la respuesta sería muy simple: porque el mismo usuario así lo dijo previamente. Y sería fácil imprimir un corto mensaje para ello.

Sin embargo, los sistemas más complejos necesitan métodos más

complejos, porque normalmente el usuario no va a preguntar sobre la validez de alguna afirmación que él mismo haya hecho. Preguntará sobre alguna conclusión intermedia obtenida por el propio sistema. En este caso, la técnica consiste en retroceder por la cadena de razonamientos que ha conducido a esa conclusión intermedia, indicando las reglas que se han aplicado y la información que se ha utilizado. La aplicación evidente de esta facilidad se encuentra de nuevo en el desarrollo del programa, en el momento en que el sistema comete un error (haciendo, por ejemplo, una pregunta improcedente), y el experto humano quiere saber qué es lo que ha hecho para llegar hasta ahí. Si el «¿por qué?» es la facilidad de instantánea, el «¿cómo?» es la de trazado.

Es muy fácil percatarse de que mejoras como las que ofrece TEIRESIAS pueden ser muy útiles para el desarrollo de un sistema experto, pero siguen siendo de utilidad incluso si el sistema funciona perfectamente.

Porque, si el sistema funciona bien e incluye la posibilidad de explicar sus acciones en términos próximos al lenguaje natural, entonces tenemos un sistema que puede utilizarse para formar a otras personas en su área de experiencia.

En resumidas cuentas, si tenemos un estudiante de medicina que no sabe mucho de diagnóstico y un sistema en ordenador que sí sabe, podemos hacer que uno se siente frente al otro y que interactúen. Imaginamos que una técnica de este tipo podría reducir enormemente la incidencia de ataques de apoplejía entre aquellos que hasta ahora vienen cargando con la pesada tarea de educar a los jóvenes.

Y, en efecto, se ha ensayado este enfoque.

Se ha desarrollado un programa llamado GUIDON, que funciona conjuntamente con MYCIN, con el fin de explotar el conocimiento de éste sobre diagnosis y aplicarlo a la enseñanza. Y, usando asimismo GUIDON, se ha adaptado a MYCIN el conjunto de reglas de PUFF (véase el próximo apartado), de modo tal que la enseñanza puede extenderse también al campo de los trastornos pulmonares.

Desde luego, hubiera sido posible utilizar esos sistemas expertos, tal como son, para la enseñanza. Pero las modificaciones permiten al sistema actuar de una manera más estrechamente ligada al comportamiento del estudiante, con un grado de interacción mayor del que habría si el estudiante se limitara a permanecer sentado mirando a la pantalla hasta que llegase la hora de irse a casa.

Podría usted pensar que, después de todo el trabajo que ya se ha

hecho en el campo de los sistemas expertos para diagnóstico médico, ya no harían falta, en realidad, los médicos. Y aunque algo de verdad hay en esta suposición, ninguna autoridad sanitaria de ningún país ha permitido hasta ahora que ningún sistema experto ejerza la medicina, pese a los informes de que tales sistemas funcionan tan bien como los expertos humanos. Y suponemos que, si es obligatorio tener un experto (humano) con permiso para ejercer, es más económico dejar simplemente que sea él quien diagnostique.

Lo cual es una pena, ciertamente. La idea de un sistema experto con un dispensario de píldoras bajo el teclado es atractiva. A menos, desde luego, que sea uno mismo el que está enfermo. La cosa entonces es muy diferente.

9.2 PUFF: TRASTORNOS RESPIRATORIOS

Después de haber comentado MYCIN, consideremos lo siguiente:

¿Qué ocurriría si cogiésemos MYCIN y lo sacudiéramos sobre la papelera hasta que cayera todo el conocimiento específico de su dominio de experiencia y sólo quedase el mecanismo básico de razonamiento?.

La respuesta es que tendríamos EMYCIN («Empty MYCIN», o MYCIN vacío) (*), que sería un sistema experto de propósito casi general, y que temporalmente, no sería experto en nada. Y eso es lo que hicieron los científicos de Stanford. Y, obtenido el MYCIN vacío, procedieron a rellenarlo con otra cosa.

Esa Otra Cosa fue un conjunto de unas 50 reglas relativas a trastornos pulmonares, que, una vez introducidas en EMYCIN, dieron lugar a PUFF, apropiado nombre de un programa para diagnosticar trastornos pulmonares.

La idea es que, cuando el asustado paciente entre en el consultorio, sople en una máquina. En ello no hay nada nuevo: la máquina registra, simplemente, el volumen de aire que expira el paciente y la velocidad del aire conforme sopla. De este registro, un médico puede deducir algún tipo de diagnóstico sobre la condición del paciente.

(*) (N. del T.) En realidad, según los creadores de MYCIN, la letra «E» procedería de «Esencial». El nombre de MYCIN proviene del sufijo inglés común a muchas drogas antibacterianas. Los científicos de Stanford recomiendan no confundir EMYCIN con E-MYCIN, un medicamento registrado por Upjon Corp.: «Ni EMYCIN se puede ingerir, ni E-MYCIN cargarse en la memoria de un ordenador». (*cf.* B.G.Buchanan y E.H. Shortliffe: «Rule-based expert systems», Addison-Wesley, 1984, p.xvii).

Podría, por ejemplo, estar normal o enfermo, y, en el segundo caso, la enfermedad podría ser bronquitis, o enfisema, o muchas otras cosas.

Sea lo que sea, la idea consiste en introducir los datos en PUFF y que éste deduzca un diagnóstico.

Para empezar, hay que señalar que PUFF no recibe los datos directamente de la máquina en la que sopla el paciente. Desde luego, podría haberse modificado para que fuera así, pero no se hizo.

Lo que ocurre es que la máquina le presenta al médico varios elementos de información, más o menos importantes, sobre la respiración del paciente. El médico también dispone de algunos elementos de información generales, más o menos importantes, sobre el paciente. Por ejemplo, su sexo, edad y hábito de fumar.

Y a partir de aquí, puede usted olvidarse de la máquina en la que se sopla y concentrar su atención estrictamente en el ordenador, preparado y con su sistema experto cargado. Porque todo lo que tenemos ahora es una lista de variables y ciertos valores asociados con ellas, y el ordenador tiene que establecer un diagnóstico.

Como ensayo, se le presentaron a PUFF 150 conjuntos de datos relativos a otros tantos pacientes para ver cómo funcionaba, con el resultado de que PUFF hizo los mismos diagnósticos que un médico en el 90 por 100 de los casos.

Ahora bien, en este momento, es posible apreciar que éste es el tipo de cosas que podrían hacerse con nuestro propio sistema experto descrito anteriormente.

Podríamos haberle dado todas las variables a considerar y todos los resultados posibles (salud perfecta, bronquitis, etc.). Podríamos luego haberle introducido los 150 conjuntos de datos para una sesión de entrenamiento y haber dejado que el experto desarrollase sus propias reglas para elaborar diagnósticos. Quizás encontraríamos que el sistema acertaba a veces al hacer diagnósticos posteriores.

Sin embargo, sería bastante injusto para con los científicos de Stanford sugerir que ambos sistemas son completamente idénticos, porque no lo son.

Para empezar, PUFF no elabora sus reglas por sí mismo. Los científicos de Stanford se reunieron con otros del Centro Médico del Pacífico, que fueron los que les dijeron como hacer diagnósticos, y este enfoque es, con mucho, el más habitual en los sistemas expertos actual-

mente en uso. Una razón para ello puede ser que a los médicos no les guste que el programa haga demasiadas cosas por su cuenta. Pero una razón más probable es que normalmente se puede construir un sistema más eficaz, si se conoce previamente cómo tiene que proceder.

Por tanto, los médicos propusieron un conjunto de reglas mediante las cuales se puede realizar el diagnóstico. Y los informáticos introdujeron esas reglas en PUFF. Dicho así, parece muy fácil, y, de hecho, los informáticos de Stanford no se pronunciaron en cuanto a que hubiera sido difícil (*).

En cuanto a la forma de las reglas, vale la pena hacer algunas observaciones. De entrada, esas reglas, como las de MYCIN, están en el formato IF...THEN... Así, en un lenguaje no estrictamente médico, podríamos tener: IF (el paciente respira con dificultad) AND (fuma 200 pitillos al día) AND (no para de toser) THEN (tiene una tos de fumador).

De este modo (dejando aparte lo que contienen específicamente los paréntesis) es como muchos investigadores describen a sus sistemas expertos: como un conjunto de reglas IF...THEN... Esta forma de describir las cosas tiene una ventaja fundamental: que cualquiera que utilice ordenadores sabe lo que significa el IF...THEN... Usted mismo puede escribir un programa BASIC en términos de IF...THEN... (Y no hay nada de vergonzante en ello; de hecho, PUFF es uno de los pocos sistemas expertos que se han reprogramado en BASIC).

Pero también hay que evitar el caer en la trampa de que es exactamente así como se debe proceder. La sentencia IF...THEN... no es más que una proposición, una sentencia de la lógica. Y hay muchas maneras de establecer la misma sentencia sin utilizar en absoluto las palabras IF...THEN...

Por ejemplo, nuestro sistema experto no guarda sus reglas en términos de IF...THEN..., pero contiene exactamente la misma información

---

(*) (N. del T.) Más bien al contrario. Concretamente, según los informes, se necesitaron menos de 50 horas de interacción con un experto y menos de 10 semanas-hombre de trabajo de los ingenieros de conocimiento (*cf.* E.A.Feigenbaum: «Themes and case studies of knowledge engineering», en D. Michie (ed.): «Expert systems in the microelectronic age», Edinburgh Univ. Press, 1979, pp. 3-25). Independientemente de esto, el grupo de Stanford siguió trabajando en el perfeccionamiento de PUFF en varios sentidos. Uno de ellos consistió en programar una versión en BASIC que recoge directamente los datos del aparato de respiración (espirómetro), y que se utiliza rutinariamente en la práctica clínica en el Centro Médico del Pacífico, en un PDP-11. Otro, el ir ampliando la base de conocimiento para cubrir un espectro más amplio de enfermedades pulmonares; la última versión de PUFF tiene unas 250 reglas. (*cf.* Buchanan y Shortliffe, *ibid.*, pp. 393, 395 y 692).

que si lo hiciese. El experto genera sus propias reglas, y siempre que las aplica realiza, en realidad, la misma operación lógica que si tuviera una sentencia IF...THEN... cuyas condiciones correspondieran a las reglas que ha desarrollado.

La terminología asociada con las reglas de los sistemas expertos es bastante amplia. Generalmente se llaman «reglas de producción», porque pueden producir un resultado o una conclusión. Frecuentemente, la primera parte de la sentencia, la que sigue a IF, se llama «antecedente», y la segunda parte, la que sigue a THEN, «consecuente». Pero también puede usted llamarles «hecho» e «hipótesis», «aserción» y «deducción», «variables» y «resultados», o lo que más le guste. La única buena razón para estandarizar la terminología es que otras personas puedan entenderle, y en un campo bastante nuevo no hay aún una estandarización suficiente de los términos que permita dar muchas orientaciones en cuanto a qué es lo que van a interpretar otras personas.

Inicialmente, PUFF sólo incluía 55 reglas, lo cual es bastante alentador, porque podemos esperar que la mayoría de los ordenadores puedan almacenar 55 reglas, y, por tanto, es interesante pensar que la mayoría de ellos puede contener algo útil.

Ahora, antes de entrar a considerar lo que hace PUFF con sus reglas, tenemos todavía que decir algunas cosas más. Para empezar, que PUFF no funcionó muy bien al principio. Típica queja sobre la mayoría de los programas en desarrollo, que debería ser alentadora para el aficionado. El problema era que las reglas de diagnóstico que dieron inicialmente los médicos no eran perfectas. Sencillamente, las reglas que comunicaron a la gente de Stanford no eran lógicamente capaces de diagnosticar todo cuanto podía presentarse. Y aquí es cuando entra en escena el Ingeniero del Conocimiento.

PUFF genera un diagnóstico equivocado. Los médicos dicen que es un mal diagnóstico. Por tanto (pregunta el Ingeniero del Conocimiento), ¿cuál debería haber sido?, ¿por qué?, ¿cuál de las reglas actuales ha fallado?, ¿qué nueva regla hay que añadir?

Los médicos piensan sobre ello y proponen algunas sugerencias que se añaden a la base de reglas de PUFF. Este proceso consistente en manipular las reglas hasta que la cosa funciona, parece un procedimiento trivial, e intelectualmente quizás lo sea. Pero no es un problema trivial en lo que se refiere a la construcción de un sistema experto. De hecho, es uno de los problemas sobre los que más se escribe.

A nivel mecánico, tiene que haber una facilidad que permita arreglar el sistema para que funcione mejor, un modo fácil de modificar y añadir

reglas. Y ello es así porque casi todos los sistemas necesitan modificaciones de vez en cuando, y el peligro está en que, si es difícil hacerlo, no se hará tan a menudo ni tan cuidadosamente como sería conveniente. Una observación práctica, pero importante.

A nivel más abstracto, podría pensarse que un grupo de médicos consiguiera dar de entrada las reglas correctas. Y, de nuevo, es importante señalar que no pueden. El experto humano que trabaja en un problema suele tener alguna idea sobre cómo lo resuelve, pero casi todos están de acuerdo en que, inicialmente, no tienen una idea muy clara. Desde luego, no lo suficientemente clara como para que un ordenador pueda imitarlos.

Lo que tiene lugar es una especie de proceso de aprendizaje mutuo en el que tanto el programa de ordenador como el experto humano descubren cómo éste ha venido trabajando durante años. Ayudado por los esfuerzos del Ingeniero del Conocimiento, que está entre el experto y el programa, éste se va perfeccionando conforme se van desenmarañando gradualmente los métodos que utiliza exactamente el experto.

Todo ello pone de relieve el problema básico de cómo introducir el conocimiento del experto en la máquina. En muchos sentidos, no es estrictamente un problema informático. Es el problema de encontrar un experto y comprender lo que dice suficientemente bien como para poder escribir un programa que haga lo que él hace. Se parece, más que nada, al análisis de sistemas. Pero no es un problema que pueda ignorarse en absoluto, porque con frecuencia se considera que constituye el mayor (¿o el menor?) cuello de botella en todo el asunto de la construcción de un sistema experto.

Nosotros, naturalmente, hemos hecho trampa. Al diseñar un sistema que construye sus propias reglas hemos eludido el problema de buscar y comprender a un experto humano, pero esto quizás haya sido a costa del producto final.

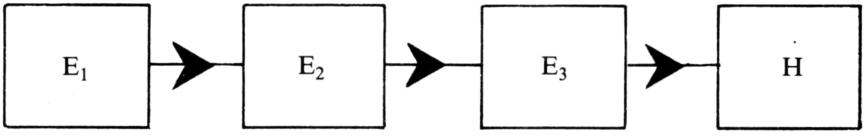
La observación importante es que, exceptuando los sistemas más sencillos, tiene que ser bastante fácil el añadir nuevas reglas y el modificar las antiguas.

Carece de importancia cómo se almacenan las reglas exactamente. Pueden estar explícitamente programadas (con el riesgo que ello conlleva de hacer que las modificaciones frecuentes sean difíciles), o guardarse como listas de antecedentes/consecuentes, como si fuesen datos. Este último método es el que, en principio, prefieren los que construyen sistemas expertos. El tratar las reglas como datos permite, en teoría, modificarlas fácilmente sin tocar al programa que las manipula.

Una vez que PUFF tiene un conjunto de reglas, es razonable preguntarse qué es lo que hace con ellas. Si se tratase de nuestro propio sistema experto, las recorrería gradualmente, sacando cuantas conclusiones pudiera. Pero PUFF y MYCIN no funcionan así, son mucho más ordenados.

En general, hay dos métodos para manejar conjuntos de reglas, que normalmente se llaman encadenamiento hacia adelante y encadenamiento hacia atrás.

El sistema que hemos descrito en la primera parte del libro utiliza encadenamiento hacia adelante, puesto que calcula lo que puede a partir de lo que tiene, y, hecho eso, sigue calculando en base a lo que acaba de obtener. El encadenamiento hacia adelante, como su nombre indica, su pone avanzar por las reglas, empujado siempre por las características presentes en los datos que se han dado.

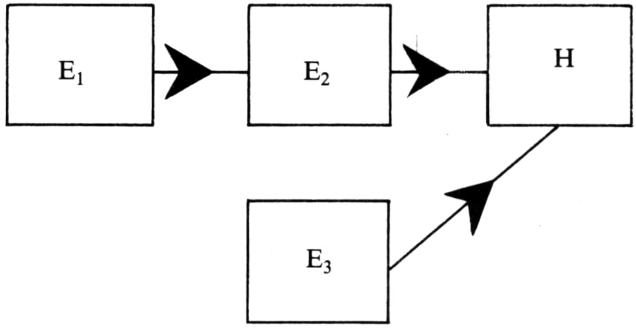


IF E_1 THEN: IF E_2 THEN: IF E_3 THEN: H, de modo que hay que demostrar E_1, E_2 y E_3 para poder demostrar H, como indica la figura 9.1. Así es exactamente como se procede con una estrategia de encadenamiento hacia adelante, o impulsada por los datos: se le da al sistema E_1, tras lo cual se le da E_2, y después E_3, y entonces ya puede deducir H.

Con una estrategia de encadenamiento hacia atrás, o impulsada por los objetivos, el sistema primero considera H y pretende deducir si es cierta o no. Mirando hacia atrás, ve que para ello necesita conocer E_3, y para E_3 necesita E_2, y para éste, E_1. Por tanto, pide información sobre E_1, tras lo cual, sigue hacia adelante.

La diferencia aparece más claramente cuando hay un gran número de conclusiones diferentes (objetivos, H) a los que puede llegar el sistema, y una diversidad de caminos para cada objetivo.

Fig. 9.1
Encadenamiento hacia adelante



En la figura 9.2, H se demuestra más fácilmente preguntando primero por E_3 que empezando por E_1.

En este caso, un encadenamiento hacia adelante empezaría con E_1 para demostrar E_2 y demostrar finalmente H. El encadenamiento hacia atrás empezaría con H, y luego miraría a E_2 o E_3. Al ver que E_3 es el camino más corto, obtendría E_3 para demostrar H, ignorando E_1.

Fig. 9.2
Razonamiento alternativo

Por su parte, el encadenamiento hacia atrás consiste en retroceder (¡sorprendente!), y permite un comportamiento mucho más dirigido por objetivos. Comienza planteando una hipótesis (¿sufre el paciente de bronquitis?) e intenta resolver si es cierta o no. Para ello, busca una regla que tenga entre sus resultados bronquitis y mira cuáles son sus antecedentes (es decir, ¿qué daría como resultado bronquitis?). Si tiene datos acerca de esas variables, puede llegar a una conclusión sobre la posibilidad de bronquitis, y, si no los tiene, toma una de las variables y trata de conseguir un dato para ella. Y eso puede hacerlo, obviamente, pidiendo al operador que proporcione ese dato, o también retrocediendo por las reglas hasta encontrar otra que, si se satisface, proporcione el dato como uno de sus resultados.

El método es esencialmente recursivo, y (por si acaso está usted pensando en ello), en BASIC no es fácil programar técnicas recursivas.

Pero, conceptualmente, si piensa usted en su propio sistema, viene a ser como acudir al último nodo del sistema (o a algún nodo que tenga un

resultado etiquetado como un estado-objetivo) y tratar de conseguir valores para todas sus entradas retrocediendo a lo largo de la cadena de nodos precedentes que dan entradas a este nodo final.

El argumento a favor del encadenamiento hacia atrás es que le da al sistema un sentido de finalidad. No se trata de que busque cuanto pueda de un modo arbitrario, sino de determinar específicamente si algunas cosas importantes son ciertas o no.

Teniendo un conjunto de reglas bien formulado, es muy discutible si tiene alguna importancia el método que se utilice. Pero si por casualidad llega usted a tener un sistema experto cuyo conocimiento se ha ido acumulando durante un largo período de tiempo, y que contiene, digamos, un amplio surtido de reglas con distintos grados de utilidad y una gran variedad de resultados con distintos grados de interés, puede ser ventajoso tener un sistema que llegue al final lo más rápidamente posible y no un sistema que trate de establecer la verdad o falsedad de todo lo imaginable cada vez que se utiliza.

Como consideración final acerca de PUFF, podríamos preguntarnos qué beneficios se consiguieron con el programa. ¿Por qué no dejar un asunto como éste a los expertos humanos?

La respuesta obvia es que en algunos campos hay cierta escasez de expertos. Pero, en muchos sentidos, esta respuesta es demasiado obvia.

A fin de cuentas, podrían haber escrito un libro o un manual sobre el tema, y dejar las cosas así. ¿Por qué molestarse en escribir un programa de ordenador?

Bien, hay algo de cierto en esta sugerencia, y quizás haya también algo de cierto en la sugerencia de que los sistemas expertos se escriben, simplemente, porque su escritura es interesante.

Pero conviene recapacitar en que las primeras versiones de PUFF, lo mismo que las de otros sistemas expertos, no funcionaron muy bien. Indudablemente, los expertos humanos, cuando formularon el conjunto inicial de reglas, creían que sabían cómo trabajaban, pero parece ser que no lo sabían tan bien. Por tanto, si hubiesen escrito un libro, ¿qué valor habría tenido?

En cierto sentido, podría generalizarse esta observación a los ordenadores en su totalidad: que realmente son grandes Máquinas de Pensamiento. Si usted imagina algún modo de hacer las cosas, puede probar a programarlo para un ordenador. Si el programa no funciona muy bien, entonces puede estar seguro de que ha sido incapaz de expresar lo que

quería que hiciese, lo que implica que realmente usted mismo no comprende bien el asunto. O no lo suficientemente bien.

Sólo cuando el programa funcione bien puede usted estar seguro de comprender totalmente el problema, lo cual hace que el ejercicio de programar no sea sólo un ejercicio para educar al ordenador: es también un ejercicio para educarse a sí mismo.

Se dice a veces que una persona tiene que comprender realmente a fondo una materia para poder enseñarla. De manera similar, hay que tener una buena comprensión de un tema para poder enseñarle la esencia de ese tema a un montón de *chips*.

9.3 DENDRAL: ESTRUCTURAS QUÍMICAS

Considerando que los sistemas expertos son la última novedad, puede que resulte sorprendente enterarse de que DENDRAL se remonta a 1965. En 1965 había ordenadores, desde luego. Pero era la época del transistor, de la cinta de papel y de la tarjeta perforada, y no la del *chip* y la pantalla.

Todo ello no tiene nada que ver con DENDRAL en sí mismo, excepto la observación de que debe ser el sistema experto más antiguo y más conocido del mundo. O, al menos, el sistema más antiguo que se haya anunciado como «experto».

Como MYCIN y PUFF, procede de la Universidad de Stanford, y es el producto de un esfuerzo conjunto entre los informáticos y un grupo de expertos humanos, en esta ocasión en el campo de la química. La idea básica es la siguiente:

Cuando un químico ha preparado una sustancia, generalmente quiere saber cuál es su estructura química, y dispone de varias formas para averiguarlo. En primer lugar, puede hacer algunas conjeturas inteligentes, utilizando su propio conocimiento experto. Luego, puede poner algo de la sustancia en un espectrómetro y estudiar las rayas espectrales resultantes para refinar sus conjeturas iniciales. Frecuentemente, ello le permite concretar la estructura exacta de la sustancia, y todos tan felices. El problema es que todo esto lleva tiempo, y exige un poco de experiencia de naturaleza humana. Y aquí es donde entra en juego DENDRAL: es un intento de automatizar el proceso de deducción de la estructura química correcta.

A grandes rasgos, el proceso es idéntico al utilizado en nuestro pro-

pio sistema o en PUFF. El usuario le presenta a DENDRAL cierta información sobre la sustancia junto con los datos espectrométricos (infrarrojos, resonancia magnética nuclear y espectrometría de masas) y DENDRAL sugiere un «diagnóstico», en el sentido de una estructura química apropiada. Pero basta profundizar un poco en el funcionamiento de DENDRAL para descubrir que las diferencias son tan grandes que hacen que las similitudes desaparezcan casi por completo. La causa de ello son dos hechos principales. Uno, que la estructura de las sustancias químicas no puede describirse fácilmente con palabras. El otro, que el número de estructuras posibles es enorme: existen millones de posibilidades.

Empecemos por el primer hecho, la descripción de las estructuras.

Si tiene usted unos conocimientos mínimos de química, habrá visto dibujos de estructuras químicas. Esos gráficos que muestran a los átomos y sus enlaces. Puede que haya visto usted, por ejemplo, un anillo de benceno, y en ese caso, ya sabe de qué se trata. Si sólo existieran el anillo de benceno y algunas otras estructuras, entonces podríamos referirnos a ellas por un nombre y proceder del mismo modo que si estuviésemos diagnosticando la presencia de, por decir algo, bronquitis, ante la evidencia de ciertos datos. El problema es que hay tantas posibilidades diferentes de interconexión que no es factible darle un nombre a cada una. La única forma de proceder es dibujándolas. Y algunas de ellas son tan complejas que los dibujos distan mucho de ser sencillos. La forma de abordar este problema es describiendo cada estructura en términos de un grafo con una variedad de nodos y de enlaces. De este modo, cualquier estructura posible puede describirse en un «lenguaje» universal.

Eso está muy bien. Pero hay que hacer una observación: que ese «lenguaje» no puede utilizarse para describir, por ejemplo, diagnóstico médico; sencillamente, el lenguaje de la teoría de grafos no es adecuado para ello. A veces se dice que la idea de un sistema experto consiste en proporcionar un programa de razonamiento de propósito general. Este programa podría hacerse experto en cualquier área que escogiésemos, simplemente desenchufando un conjunto de reglas sobre un tema y enchufando otro conjunto de reglas sobre otro tema. DENDRAL es un ejemplo muy bueno de hasta qué punto este ideal es difícil de alcanzar, porque destaca el hecho de que un lenguaje descriptivo puede ajustarse a un problema, pero no a otro. Sería sumamente molesto (si es que no es prácticamente imposible) forzar la descripción del diagnóstico médico en el lenguaje de la teoría de grafos, o forzar la descripción de las estructuras químicas en el lenguaje de la medicina. Y en la medida en que las diferentes áreas se ajustan mejor a diferentes lenguajes descriptivos, vemos que es necesario adoptar un enfoque especializado para cada área.

El segundo hecho concierne al número de estructuras posibles. En el caso del diagnóstico médico, pueden guardarse en memoria todos los resultados posibles, y lo mismo ocurre en otros campos. Pero cuando el número de resultados posibles llega a ser del orden de millones, ello, evidentemente, es imposible. Por tanto, el problema ahora es el de escoger la estructura correcta cuando la máquina no «conoce» detalladamente cuáles son las estructuras posibles.

La forma en que DENDRAL resuelve este problema, consiste en generar estructuras posibles en tiempo de ejecución e irlas comprobando para ver si son las correctas. Si no tuviese restricciones en su comportamiento, podría generar todas las estructuras químicas posibles, pero de nuevo tropezamos con el mismo problema: que hay demasiadas estructuras. El sistema aborda este problema generando sólo un pequeño subconjunto de todas las posibles.

En efecto, puede considerarse que DENDRAL tiene dos partes. Es algo así como dos sistemas expertos distintos en uno solo.

La primera parte contiene un conjunto de reglas para generar estructuras químicas posibles. Los datos de entrada a esta parte consisten en una serie de afirmaciones hechas por un químico, que dan algunas pistas sobre las estructuras más probables en cada caso. En cierto modo, esto es muy parecido a los sistemas que hemos considerado antes.

Pero la salida de este primera parte no es una respuesta única. Normalmente, es todo un conjunto de estructuras posibles; el programa es incapaz de decir cuál es la correcta.

La segunda parte de DENDRAL considera entonces cada una de esas estructuras y utiliza un segundo sistema experto para calcular, para cada una, cuáles serían los resultados espectrométricos si esa sustancia existiera realmente y se la colocara en un espectrómetro. Las entradas a este segundo sistema experto son las estructuras generadas por el primero, las reglas que utiliza se obtuvieron de químicos reales, que describieron el comportamiento de un espectrómetro, y las salidas son las respuestas simuladas del espectrómetro. Y en este punto hay que recordar que una entrada adicional a DENDRAL consiste en las lecturas reales del espectrómetro que provienen de la sustancia que se está considerando. DENDRAL compara los resultados de sus simulaciones con los reales, para ver si son iguales. Si lo son, la estructura simulada puede ser la correcta, y si no lo son, la descarta y genera otra estructura.

El proceso consiste en podar continuamente el árbol para mantener en todo momento un número de posibilidades lo más pequeño posible. A diferencia de otros sistemas expertos, DENDRAL no es un «juguete». No se ha hecho para probar una teoría sobre los sistemas expertos,

sino que tiene una utilidad real para identificar estructuras químicas, y se han escrito más de dos docenas de artículos científicos utilizando los resultados de DENDRAL sobre problemas reales. Se ha dicho que, en su campo, compite con los expertos humanos. En este sentido, induce ciertamente a un optimismo, en el sentido de que puedan construirse otros sistemas útiles en otros campos. Pero es discutible que estos otros sistemas puedan hacer uso de los métodos de DENDRAL.

Desde luego, la idea general de reducir la extensión de la búsqueda mediante algunas restricciones iniciales, generar soluciones posibles y contrastar esas posibles soluciones con algún otro criterio, podría aplicarse en otros campos.

Pero el programa concreto de DENDRAL, en el que las soluciones se describen en términos de teoría de grafos y se contrastan a través de simulaciones de espectrometría, podría ser difícil de transportar a otros campos.

Y ello conduce a una observación bastante general: los sistemas expertos sencillos pueden convertirse con relativa facilidad para que funcionen en otro campo, pero su utilidad es reducida en cualquiera de los campos. Y los sistemas que son verdaderamente útiles lo son gracias a su complejidad y a su especialización. Al hacer que resuelvan un conjunto muy particular de problemas, resultan ser altamente especializados en ese área de problemas, y se hace difícil, si no imposible, adaptarlos a otras áreas.

Como una ilustración adicional de este aspecto, consideremos META-DENDRAL. Ya sabemos que DENDRAL tiene un sistema para simular resultados espectrométricos a partir de una determinada estructura química. Pues bien, META-DENDRAL es el sistema experto que elabora esa simulación. El problema era: ¿Cómo se simulan los resultados espectrométricos?; ¿qué reglas ha de tener un sistema experto para poder hacerlo? Y para descubrir eso se construyó META-DENDRAL. Funciona, aproximadamente, lo mismo que DENDRAL, puesto que puede generar toda una serie de reglas posibles, que va aplicando a los datos de entrada para ver cuáles de ellas pueden explicar los resultados.

Concretamente, recibe como datos cierta salida del espectrómetro y una descripción gráfica de la sustancia que da lugar a esa salida. Y entonces empieza a generar una serie de reglas que podrían ser aplicables a esos datos del espectrómetro, aplicando cada regla y comprobando si su salida es la misma que la de la estructura dada. Si no lo es, sigue generando otra regla, y, si lo es, entonces se trata de una posible regla.

Trabajando sobre un gran número de ejemplos, META-DENDRAL

llegó a generar un conjunto de reglas que permiten mostrar la salida probable del espectrómetro para cada estructura dada.

En cierto modo, este funcionamiento es similar (aunque lejanamente) al del sistema descrito en la primera parte de este libro, en el que le dábamos al experto una serie de ejemplos con resultados conocidos y él elaboraba por sí mismo un conjunto de reglas. Y todo ello está muy estrechamente ligado con uno de los principales problemas del campo: que cuando se comienza a construir un sistema experto es frecuente que no se conozcan realmente las reglas que hay que usar, y se necesita algún modo, preferiblemente sencillo, para descubrirlas.

Observábamos, al hablar de PUFF, que, con frecuencia, el conocimiento de los expertos humanos acerca de las reglas que utilizan no es muy claro. Y ello ocurre en un campo en el que realmente hay expertos humanos. Pero, ¿por qué habría de haber ningún experto humano en el campo de META-DENDRAL?. ¿Quién perdería su tiempo calculando la salida probable de un espectrómetro, dada una sustancia cuya estructura molecular ya se conoce?. Al fin y al cabo, si tenemos la sustancia, siempre podemos ponerla en un espectrómetro y resolver así el problema. Y si no tenemos la sustancia, ¿a quién le importa el tipo de salida que pudiera dar?. La necesidad de tener un experto en el campo de META-DENDRAL surgió sólo como consecuencia de la existencia del propio DENDRAL, y no había realmente ningún experto a quien recurrir, de modo que hubo que empezar a partir de cero.

El problema es interesante. La mayoría de los sistemas expertos incorporan, de una forma u otra, el conocimiento de expertos humanos. Pero un sistema que pueda adquirir su propio conocimiento es mucho menos dependiente de un cuerpo de conocimientos preexistente, y puede verdaderamente incrementar nuestro propio bagaje de conocimientos, no ya sobre casos concretos, sino sobre todo el dominio en el que opera el sistema.

Considere el ejemplo anterior de la predicción meteorológica. Es posible que usted no sepa nada sobre predicciones meteorológicas, pero ello no importa demasiado. Puede construir un sistema y entrenarlo con una serie de ejemplos reales, y, con el tiempo, el sistema captará la idea y podrá hacer las predicciones con una precisión razonable. Ahora bien, el hecho de que pueda hacerlo significa que contiene realmente un conjunto de reglas que son mejores que las que usted pudiera tener. Y si usted fisga dentro de su conjunto de reglas, quizás pueda aprender algo sobre predicción meteorológica. Posiblemente sea un ejemplo trivial, pero el principio es interesante, y META-DENDRAL sacó buen partido de él.

9.4 PROSPECTOR: BÚSQUEDA DE MINERALES

Tiene que ser bonito poder curar al enfermo, y también puede que sea bonito ser capaz de deducir una estructura química; no cabe duda alguna de que, en realidad, lo más bonito de todo sería llegar a descubrir oro en aquellas colinas lejanas. Oro auténtico, claro. De ese que le hace a uno rico.

Sin embargo, PROSPECTOR no ayuda en una verdadera prospección para encontrar oro, porque sus inventores limitaron sus actividades a yacimientos mucho más grises, aunque también valiosos. Pero el principio es el mismo: PROSPECTOR es un sistema experto diseñado para ayudar en la búsqueda de riquezas minerales explotables comercialmente. Y, como tal, es interesante.

Imaginamos que, tradicionalmente, el experto humano en este campo carga su mula con algunas ollas y cacerolas, prepara suficientes bocadillos para resistir todo el invierno, y pone rumbo a las colinas para aplicar su experiencia. Llegada la primavera, y habiendo aplicado sus juicios expertos, puede volver, exhausto y tambaleante, a la ciudad, y reclamar legalmente sus derechos sobre el territorio aurífero. Tras lo cual, normalmente, un facineroso lo mata a tiros, quedando su hija para vengarle y recuperar el oro. El resto de la historia es bastante familiar (la hija, con la ayuda de un amigo que le echa una mano, recupera el oro y vive feliz el resto de sus días).

Pero la verdadera pregunta, que jamás encuentra respuesta satisfactoria en los relatos, es: ¿qué es lo que realmente hacía el experto cuando estaba en las colinas?

Así comienza la historia de PROSPECTOR. Porque la primera fase consistió (como con todos los sistemas expertos) en descubrir qué es lo que hacen los prospectores expertos cuando están buscando oro (o algún depósito más gris, pero también valioso). Y la respuesta, en general, es la habitual serie de inferencias y deducciones, unas exactas, otras probabilísticas, a partir de las cuales el experto se forma un juicio sobre el asunto.

Por si está usted preguntándoselo, el método no consiste en apuntar con el ordenador a una vasta región de terreno y preguntarle dónde hay que buscar oro; ello sería poco preciso para un ordenador. Consiste en especificar una posición exacta sobre la que se conocen ciertos hechos y pedir al ordenador que dé una estimación de la probabilidad de que en esta posición haya cierto depósito. En cierto sentido, es un poco como el diagnóstico médico. En éste, se le presenta al experto (humano o arti-

ficial) un paciente y se le pregunta qué es lo que va mal en ese paciente; no se le pregunta al experto sobre diagnóstico de pacientes en general.

Y, en cierto sentido, el buscador o prospector puede proceder como hace el médico. Igual que, por ejemplo, MYCIN, PROSPECTOR contiene un gran número de reglas relativas a las diferentes cosas que podrían observarse y a las cosas que podrían deducirse de ellas. Entonces, procede mediante encadenamiento hacia atrás: haciendo la hipótesis de que cierto resultado sea el correcto, y trabajando con las reglas hacia atrás para ver si se puede justificar ese resultado.

Sin embargo, a diferencia del diagnóstico médico, el resultado no es una simple afirmación sobre el asunto.

Veamos: dado un paciente del que se sospecha que está enfermo, lo que uno quiere saber es qué enfermedad tiene. Hay que escoger uno de entre varios resultados. No importa demasiado cuál sea exactamente la probabilidad de que el diagnóstico sea correcto. Se trata simplemente de obtener el diagnóstico más probable.

Contrastemos eso con la prospección mineral. En una situación determinada, cualquier mineral puede encontrarse en ciertas cantidades, aunque sean minúsculas, de modo que la conclusión de que hay cierto mineral no es muy útil. Lo que el usuario de un sistema de prospección quiere saber es qué cantidad de mineral hay. Quizás la cantidad exacta no importe demasiado: sería suficiente con poder saber que hay «mucho». Pero la probabilidad de que haya «mucho» es muy importante. Una simple respuesta sí/no no sirve cuando el coste de excavar el terreno para comprobar la opinión del experto es alto.

El tema del análisis de riesgos tiene alguna relación con el asunto, desde el momento en que hay un coste asociado con una decisión falsa.

Suponga que tiene usted un paciente y que diagnostica que padece alguna enfermedad. El diagnóstico podría ser correcto, en cuyo caso, al tratarlo para esa enfermedad, estaría haciendo lo que hay que hacer. Muy bien. Hay cierta probabilidad de que el diagnóstico sea falso, pero normalmente no es muy perjudicial para el paciente que se le trate de una enfermedad que no tiene. (No obstante, nunca ampute un brazo porque se lo aconseje un ordenador).

La esencia del diagnóstico médico es que es importante descubrir y tratar cualquier posibilidad, y generalmente no importa demasiado si resulta que la posibilidad, de hecho, no se da.

Con la prospección mineral la situación es algo diferente. Ob-

viamente, si es usted una persona a la que suele acompañarle la suerte, excavaría en todos los sitios donde pudiera encontrarse oro. Pero, en cierta medida, es más importante no buscar oro en los sitios donde no lo hay.

Excavar el terreno consume tiempo y dinero, y, en general, mientras se está excavando en un sitio no se está excavando en otro.

El truco consiste en encontrar el sitio más probable y excavar en él, y es por eso por lo que aquí es importante tener una estimación precisa de las probabilidades, lo que no ocurría en el caso del diagnóstico médico. No es que el oro sea más importante que la salud (y aunque lo sea), es sólo que las condiciones en las que ha de basarse la acción son diferentes.

Dicho esto, es interesante ver cómo maneja PROSPECTOR el asunto de las probabilidades para obtener sus conclusiones, porque, aunque son discutidos, sus métodos son los mejor conseguidos de todos los sistemas expertos.

Los casos más sencillos corresponden a las reglas que expresan relaciones lógicas. Son reglas del tipo IF x THEN z, en las que z se deduce necesariamente de x. Ahora bien, esto es fácil, y sigue siendo fácil aunque se asocie una probabilidad con x, porque puede razonarse que si la probabilidad de x es p, entonces la probabilidad de z es también p.

Pero, en general, si x tuviese un único argumento, la regla sería demasiado trivial. Usualmente, x se sustituirá por un término más complejo, como, por ejemplo, (x Y y), o (x O y).

En el caso de relaciones formadas con la conjunción Y, si cada elemento individual tiene una probabilidad asociada, PROSPECTOR toma el mínimo de los dos valores y asigna esta probabilidad mínima al resultado. Es decir, si la probabilidad de x fuese 0.1 y la de y fuese 0.2, la probabilidad de z sería 0.1. Es fácil percatarse de por qué se ha elegido este método: para que z sea cierta, es necesario que tanto x como y lo sean, lo cual es un requisito fuerte, y por ello se toma el valor mínimo.

Por otra parte, en el caso de elementos unidos por la disyunción O, se toma el valor máximo, porque tanto x como y son causa de z, lo cual representa un requisito débil.

Realmente, este método no está totalmente libre de críticas. (Puede usted echar de nuevo un vistazo al apartado sobre probabilidades). Para dar una idea del error, si x e y son variables independientes entre sí, entonces $P(x\ Y\ y) = P(x)P(y)$, es decir, la probabilidad resultante es el pro-

ducto de las dos probabilidades. Si x e y no son independientes, sino que están perfectamente correlacionadas una con otra, entonces P(x) = P(y), y P(xYy) = P(x) = P(y). Y, en general, con variables parcialmente correlacionadas, la verdad estará entre ambas cosas(*).

Del mismo modo, si las variables fueran independientes, tendríamos P(x O y) = P(x) + P(y)—P(x)P(y), y si fueran totalmente correlacionadas, P(x O y) = P(x) = P(y). Y el valor real estará comprendido entre ambos si las variables son parcialmente correlacionadas.

Así, PROSPECTOR sigue un método algo arbitrario, pero aún así es un método que da respuestas adecuadas.

El siguiente caso, más interesante, es aquél en el que aparece una serie de aserciones (reglas), cada una de las cuales contribuye algo a la probabilidad de alguna hipótesis.

Por ejemplo, podríamos decir que si hubiese oro esparcido por el suelo en grandes trozos, entonces habría oro en las colinas con probabilidad 0.9.

En sí misma, esta sentencia viene a ser igual que algunas de las que hemos visto anteriormente. La diferencia está en que en este caso realmente sólo nos importa una hipótesis (que hay oro en las colinas), y puede haber un gran número de sentencias que tengan relación con esa hipótesis, ya sea apoyándola o contradiciéndola, y el problema resulta ser cómo llevar la cuenta de todas las probabilidades.

El método que utiliza PROSPECTOR consiste en una aplicación ingeniosa de la fórmula de Bayes (veáse el apartado 2.5) para evaluar las probabilidades *a priori* y *a posteriori* de que ocurra cierto suceso, entendiendo por suceso cualquier cosa, tal como que la hipótesis concerniente al oro y a las colinas sea cierta.

Esquemáticamente, cada hipótesis empieza con una probabilidad inicial (*a priori*) de ser cierta, P(H). Así, la probabilidad *a priori* de que haya oro en las colinas podría ser, por ejemplo, 0.1. P(H) = 0.1.

Póngase ahora en el lugar del prospector, sentado ante la pantalla y presentándole al sistema un elemento adicional de evidencia. Obtenido

---

(*) (N. del T.) Recuérdese que x e y son variables aleatorias binarias. En este contexto, P(x) significa «probabilidad de que x sea cierta». Decir que x e y son independientes es lo mismo que decir que los sucesos «x cierta» e «y cierta» son independientes. Y decir que x e y son perfectamente correlacionadas, es como decir que dichos sucesos son idénticos.

este elemento de evidencia («he encontrado un trozo de oro del tamaño de un puño», por ejemplo), la probabilidad P(H) cambia a P(H:E), probabilidad de la hipótesis dada la evidencia. Por tanto, el sistema puede actualizar su antiguo valor de probabilidad con una sentencia de asignación, P(H) = P(H:E), y seguir para confirmar un nuevo elemento de evidencia.

La pregunta, entonces, es: ¿Cómo se calcula P(H:E)?.

Pues bien, el Reverendo Bayes tenía la respuesta, que no es otra que:

P(H:E) = P(H)P(E:H)/(P(H)P(E:H) + P(no H)P(E:no H))

Y lo mejor que puede hacerse, en cierto sentido, es incluir esto en el programa y olvidarse de ello. Si quiere entrar en más detalles, siga leyendo.

P(H:E) = MS*P(H)/(P(no H) + MS*P(H)), fórmula que parece algo más fácil que la anterior, y en la que MS = P(E:H)/P(E:no H).

Ambas fórmulas dicen lo mismo. MS es la medida de suficiencia, más conocida en estadística como cociente de verosimilitud. En este caso, es la probabilidad de obtener este elemento de evidencia, supuesto que la hipótesis es cierta, dividida por la probabilidad de obtener el mismo elemento, supuesto que la hipótesis es falsa.

Consideremos que la hipótesis es que hay oro en las colinas. Si es cierta, la probabilidad de encontrar un trozo de oro del tamaño de un puño es, por ejemplo, 0.3. Y si no lo es, la probabilidad del mismo suceso cae drásticamente a, digamos, 0.1. Por tanto, la medida de suficiencia tiene el valor 3.0.

Si aplicamos la fórmula para calcular P(H:E), resulta:

P(H:E) = 3*P(H)/(1—P(H) + 3*P(H)) = 3*P(H)/(1 + 2*P(H))

Y, por tanto, si la probabilidad previa de que hubiese oro en las colinas fuese del 50 por 100, P(H) = 0.5, entonces P(H:E) = 1.5/(1 + 1) = 0.75.

Es decir, la cosa mejora.

Así se obtiene un nuevo valor de P(H) para la hipótesis en cuestión, y conforme se va acumulando más evidencia, los valores de P(E:H) que van introduciéndose van modificando sucesivamente a P(H).

La historia no termina aún, porque siempre habrá algunos pesimistas que argüirán que ciertas evidencias apoyan más bien la hipótesis de que no haya oro. Concretamente, puede que usted no encuentre ningún trozo de oro del tamaño de un puño, y que quiera tener este hecho en cuenta. El cálculo es el mismo de antes, salvo que hay que considerar no-E en lugar de E, para indicar la ausencia de evidencia. Como antes, basta con ajustar P(H). Es importante observar que ahora se necesita un nuevo conjunto de probabilidades, y un nuevo cociente, porque el anterior, MS, estaba basado en la presencia de evidencia. Así, si tomamos

MN = P(no E:H)/P(no E:no H)

como nuevo cociente de verosimilitud asociado con la ausencia de un cierto elemento de evidencia (medida de necesidad), podemos calcular una nueva P(H) = P(H: no E) como antes, salvo que utilizaremos MN en vez de MS(*).

Supongamos, por ejemplo, que no hemos encontrado ese trozo de oro. La probabilidad de no encontrar un trozo, supuesto que hay oro en las colinas, podría ser, por ejemplo, 0.9, y la probabilidad de no encontrarlo, supuesto que no hay oro en las colinas, podría ser 1.0, lo cual da MN = 0.9/1.0 = 0.9. Así, la ausencia de oro por el suelo reduce la probabilidad de que haya oro en las colinas, pero no demasiado. En el ejemplo anterior, ello reduciría la probabilidad de P(H) del valor 0.5 al valor 0.47, aproximadamente.

(*) (N. del T.) Nos hemos permitido añadir al texto original las denominaciones «medida de suficiencia» y «medida de necesidad», que son las que utilizaron los diseñadores de PROSPECTOR para designar a los cocientes de verosimilitud (MS y MN). La justificación de tales nombres aparece claramente si en lugar de la *probabilidad*, P, se utiliza la *posibilidad*, S = P/(1—P) (veáse el apartado 12.2.3 de este mismo libro). En efecto, unas trasformaciones algebraicas elementales sobre las fórmulas de Bayes conducen a:

S(H:E) = MS*S(H), y S(H: no E) = MN*S(H)

Un valor MS = 1 indicaría que la presencia de la evidencia E es indiferente (S(H:E) = S(H)); cuanto mayor sea MS, tanto más *suficiente* es E para que H sea cierta, y, en el caso límite (MS infinito), E sería *lógicamente suficiente* para H (puesto que S(H:E) valdría infinito, es decir, P(H:E) = 1: la presencia de E implicaría que H es cierta). Por el contrario, valores de MN inferiores a la unidad corresponden al grado de necesidad de E: cuanto menor sea MN, tanto más *necesaria* es la presencia de la evidencia E para que la hipótesis H sea cierta, y, en el caso límite (MN = 0), E sería *lógicamente necesaria* para H (puesto que S(H: no E) = 0, es decir, la ausencia de E implicaría que la posibilidad de H sería nula.

Los diseñadores de PROSPECTOR consideraron más fácil para los expertos expresar su conocimiento mediante estas medidas (MS y MN) que no mediante la indicación de las probabilidades condicionales.

Por supuesto, si el sistema le pregunta si hay trozos de oro por el suelo, y usted le responde que sí, que los hay, entonces el sistema no tiene por qué preguntarle si es cierto que no los hay. Si tenemos los dos valores, MS y MN, es sencillamente para poder considerar los dos resultados, el negativo y el positivo, sin que ello quiera decir que el sistema tenga que molestarle con las dos preguntas.

En general, las reglas de PROSPECTOR son todas de la forma IF...THEN...(MS,MN), de modo que con cada regla se ha establecido un cociente de verosimilitud para la respuesta positiva y otro para la negativa. Estos cocientes se calculan como hemos descrito, y tienen su origen en la mente de los prospectores expertos humanos. Los diseñadores del sistema le preguntan a los expertos cosas como: «si hubiese oro en las colinas, ¿en cuánto estima usted la probabilidad de que se encontraran grandes trozos de oro?», y si no lo hubiese, «¿en cuánto estima la probabilidad de no encontrar tales trozos?»(*).

En total, serían necesarias cuatro preguntas para cubrir todas las posibilidades, que darían valores para:

P(E:H)
P(E: no H)
P(no E:H)
P(no E: no H)

En realidad, a fin de no complicar las cosas, hemos considerado un caso bastante especial: que el usuario del sistema conoce realmente la respuesta a la pregunta que plantea el sistema.

Al fin y al cabo, cualquier idiota puede saber si ha encontrado un trozo de oro del tamaño de un puño o no.

Pero en general, en la vida real, las respuestas son mucho menos seguras.

Los prospectores de otros minerales tienden a plantear preguntas bastante específicas, como: «¿se ha modificado profundamente la hornablenda en biotita?».

Y, francamente, uno comprende muy bien a cualquiera que no esté seguro del todo de la respuesta a tal pregunta.

(*) (N. del T.) Realmente, como hemos dicho antes, lo que se les pide a los expertos son las medidas de suficiencia y de necesidad.

La solución de PROSPECTOR consiste en dar a cada usuario una escala de —5 a +5 para responder. Una respuesta de +5 corresponde a un sí categórico, y un —5 es como un no seguro.

Normalmente, el usuario responderá con algún valor intermedio, y PROSPECTOR lo considera ajustando P(H) con un poco de MS y otro poco de MN mediante un procedimiento de interpolación lineal. Puede pensarse en una escala lineal con MN a la izquierda y MS a la derecha. Y con una respuesta del usuario comprendida ente —5 y +5, P(H) se ajusta según el valor M que se extrae de esa escala que va de uno a otro(*).

Este método que utiliza PROSPECTOR es bastante elegante en teoría, y, según los informes, bastante bueno en la práctica. Pero, como la mayoría de los expertos, no es tan modular en diseño como en principio parece.

La dificultad más importante aparece cuando el usuario empieza a dar respuestas que no son simplemente sí o no. Por ejemplo, el sistema podría necesitar saber la edad estimada de un trozo de roca, y la respuesta podría ser una cifra aproximada en años. Esto conduce a una situación en la que los valores de MS y MN no son valores simples. En general, los cocientes de verosimilitud son funciones de una variable, y tales funciones han de estar almacenadas en algún sitio y han de calcularse en el momento en que se da la entrada. No es un problema grave, pero supone una desviación de la modularidad estricta. De igual modo, los elementos puramente lógicos requieren una programación algo diferente de los elementos probabilísticos, que son más generales.

Como consecuencia, aunque los principios que utiliza PROSPECTOR pueden adaptarse a otras áreas(**), no es fácil adaptar el programa concreto, comentario que puede hacerse extensivo a muchos otros sistemas expertos.

(*) (N. del T.) La fórmula que se utiliza, concretamente, es la siguiente: con el valor E (entre —5 y +5) que da el usuario para la evidencia, se calcula $M = MS*(5+E)/10 + MN*(5-E)/10$, y el valor previo de la posibilidad de H, S(H), se sustituye por M*S(H).

(**) (N. del T.) De hecho, existe un sistema derivado de PROSPECTOR, llamado KAS (Knowledge Acquisition System: Sistema de adquisición de conocimiento), que viene a ser a PROSPECTOR algo parecido a lo que EMYCIN es con relación a MYCIN, y que se utiliza para construir otros sistemas expertos. Por otra parte, el sistema que se describe en el siguiente capítulo de este libro sigue bastante fielmente el mecanismo inferencial de PROSPECTOR.

Y, en buena medida, el grueso del trabajo que supuso la construcción del sistema parece que consistió en la obtención de las reglas a partir de los expertos humanos en el tema, comentario que también puede hacerse sobre la mayoría de los otros sistemas.

Pero la impresión de conjunto (que sigue siendo válida para la mayoría de los sistemas) es que, una vez que se ha interrogado a los expertos humanos y se ha escrito un conjunto de reglas, se ha terminado la parte más difícil del trabajo. Porque, una vez que se ha hecho eso, la forma en que hay que programar el sistema, resulta, hasta cierto punto, evidente. O, al menos, algunas formas resultan muy poco atractivas mientras que otras se destacan como preferibles.

Todo ello se parece un poco a la diferencia existente entre el análisis de sistemas y la programación. Si por «programación» entendemos todo el trabajo, entonces puede ser bastante complicada. Pero si sólo supone la codificación de una estructura previamente determinada mediante el análisis de sistemas, entonces es bastante sencilla. En el campo de los sistemas expertos, el gran problema radica en descubrir esta estructura global antes de comenzar con la programación real para un área de experiencia determinada.

9.5 OTROS EJEMPLOS

La dificultad para dar ejemplos de sistemas expertos es que la definición de sistema experto es suficientemente imprecisa como para poder catalogar como tal casi cualquier cosa. Ello no significa que la definición sea absolutamente inútil, sino que es flexible. Puede considerarse en sentido amplio (en cuyo caso la lista de los sistemas expertos existentes sería inacabable), o en sentido restringido (en cuyo caso no existiría casi ningún sistema experto).

Y no sólo somos nosotros quienes tenemos este problema. En el momento en que escribimos esto, la Agencia Central de Ordenadores y Telecomunicaciones del Reino Unido (CCTA: Central Computer and Telecommunications Agency) está preocupada ante la perspectiva de que Gran Bretaña se vea arrollada por los sistemas expertos (y baratos) procedentes de Japón, y ha encargado un estudio para averiguar lo que se está haciendo sobre el tema. Este estudio ha consistido (y no es nuestra intención descalificar a ninguna de las partes implicadas) en un cuestionario que comienza con una definición de sistema experto (la que hemos dado al principio del libro) y luego contiene preguntas que, dicho brevemente, le piden a los encuestados que digan si tienen uno, y, si es así, qué es lo que hace. En muchos sentidos, es un enfoque razonable, pero sería más razonable si se aplicara a algo cuya existencia fuera menos dudosa.

Por ejemplo, hacia el final de este libro encontrará usted un programa con un mecanismo de inferencia bayesiano que puede utilizarse fácilmente para diagnóstico médico. Introdúzcalo, pues, en su micro y tendrá un sistema experto, y podrá llamar a la CCTA y decírselo. Si todo el mundo hiciese tal cosa, el uso de los sistemas expertos, aparentemente, subiría como la espuma. Puede también hacer lo mismo con el sistema de la primera parte de este libro. O, también, bajando más, puede escribir un programa que haga algo (cualquier cosa) y decidir que es un sistema experto. ¿Y quién le va a decir que no lo es?

Lo importante es que preguntar a alguien si tiene un sistema experto, no es lo mismo que preguntarle, por ejemplo, si tiene un televisor.

En cualquier caso, damos a continuación una lista de algunos sistemas de los que se afirma que tienen un gran componente basado en conocimiento.

Nombre del sistema	*Propósito*
MYCIN	Diagnóstico médico
PUFF	”
PIP	”
CASNET	”
INTERNIST(*)	”
SACON	Diagnóstico en ingeniería
PROSPECTOR	Diagnóstico en geología
DENDRAL	Química
SECS	”
SYNCHEM	”
EL	Análisis de circuitos
MOLGEN	Genética
MECHO	Mecánica
PECOS	Programación
R1(*)	Configuración de ordenadores
SU/X(*)	Acústica de máquinas
VM	Medidas médicas
SOPHIE	Tutoría sobre electrónica
GUIDON	Tutoría sobre medicina
TEIRESIAS	Adquisición de conocimiento
EMYCIN	”
EXPERT	”
KAS	”
ROSIE	Construcción de sistemas expertos
AGE	”
HEARSAY III	”

AL/X	”
SAGE	”
Micro-Expert	”

(*) (N. del T.) Las últimas versiones de INTERNIST, R1 y SU/X se llaman CADUCEUS, XCON y HASP/SIAP, respectivamente.

Hay ciertas observaciones interesantes a hacer sobre esta lista. Y, sin duda, lo primero a observar es el número de sistemas expertos que sirven para construir otros sistemas expertos, y seguramente ello ayuda a percibir mejor la dificultad de la definición. Consideremos por un momento los compiladores. Son programas de ordenador cuya función es ayudar a la gente a escribir programas de ordenador. Y al permitir que la definición de sistema experto incluya a los sistemas que ayudan a la gente a escribir sistemas expertos, ¿no estamos, de algún modo, dando una definición que ciertamente es tan imprecisa como la expresión «programa de ordenador»?. Quizás no importa mucho que lo sea. Al fin y al cabo, podríamos sentarnos tranquilamente y decir: «si son útiles, ¿qué importa cómo se llamen?». Y puede que, después de todo, esa actitud sea la mejor que pueda adoptarse. De otro modo, existe el peligro de acabar siendo tan pedante como para criticar el trabajo de otras personas simplemente porque no encaja en alguna etiqueta preconcebida.

El siguiente grupo interesante de sistemas es el formado por los que se utilizan para adquisición de conocimiento. Ya hemos hecho observar que la introducción del conocimiento de los expertos en el sistema es una de las tareas más difíciles, y que la ayuda del propio ordenador para ello sería conveniente.

Y nosotros mismos tenemos nuestra propia ayuda: nuestro sistema de aprendizaje, por ejemplo es, con todo derecho, un Sistema de Adquisición de Conocimiento.

Si ha conseguido usted que funcione su propio sistema experto (el que hemos descrito anteriormente), quizás esté interesado en saber si puede funcionar en alguna de las áreas listadas arriba.

El diagnóstico médico sería sencillo. Use los síntomas como variables de entrada y las enfermedades como resultados, e introduzca algunos ejemplos en el sistema para ver cómo se las arregla (aunque para esta tarea puede que el sistema bayesiano que se describe más adelante funcione mejor).

¿El diagnóstico en ingeniería?. Sí. Usted puede tratar de construir un sistema para ver por qué su coche no arranca.

¿Geología?. Quizás. Depende bastante de sus conocimientos sobre geología. La ventaja de la medicina es que se pueden comprar enciclopedias médicas relativamente baratas que dan información sobre gran cantidad de enfermedades. Pero, desgraciadamente, no hay muchos textos sobre prospección mineral.

Cualquiera que se interese por la paleontología puede clasificar fósiles utilizando nuestro experto. Por ejemplo, un lamelibranquio tiene una concha asimétrica, mientras que un braquiópodo tiene una concha simétrica. Por consiguiente, ¿es un lamelibranquio o un braquiópodo?. La pega está en que, sabido esto, una vez que se mira la concha no se necesita para nada un ordenador. A la mente del entusiasta vendrán rápidamente ejemplos más complejos.

¿Química?. Sí. El análisis químico (a nivel de bachillerato) involucra una serie de pruebas con resultados concretos. Por tanto, podría ayudarle para algunos análisis químicos.

¿Análisis de circuitos?. También sí. Como el sistema sólo considera respuestas sí o no, podría ser que no funcionase muy bien para circuitos analógicos (en los que las corrientes varían de modo continuo), pero podría representar muy fácilmente a un proceso digital. Utilice un sistema multinodo y haga que cada nodo represente a un componente cuyas entradas valen 0 ó 1. Interconecte los diferentes nodos, y podrá tener un sistema experto para representar a toda una placa de chips. Si ciertos estados son correctos (es decir, acordes con las intenciones del diseñador), podrían estar controlados por otro nodo calculado para discernir los estados buenos de los malos. Así, el experto podría simular la operación de los chips y al mismo tiempo controlarla para ver si es satisfactoria.

Sin embargo, muchas de las aplicaciones listadas tenderían a dejar estancado a nuestro propio sistema experto, principalmente, porque éste es demasiado general para poder adaptarse a algunas de esas aplicaciones específicas.

Consideremos MECHO, por ejemplo. Se trata de un sistema que puede dar respuestas inteligentes a problemas mecánicos complicados. Supongamos, por ejemplo, que tenemos un sistema de poleas, cuerdas y pesas. En el lugar de un peso hay una cacerola vacía colgada de una cuerda. Con esta cacerola vacía, el sistema no está en equilibrio, y la pregunta es: ¿qué peso hay que poner en la cacerola para llevar al sistema al equilibrio?. Es un problema muy familiar de física elemental. Para responder a la pregunta, el sistema experto necesita comprender la disposición de las poleas y las leyes aplicables de la física, y tiene que poder utilizar este conocimiento para obtener una respuesta. MECHO

puede hacerlo. Y, en un sentido muy tosco, nuestro sistema multinodo también podría hacerlo. Utilizando cada uno de los nodos para representar una polea, las entradas podrían ser los pesos que cuelgan de esa polea, y las salidas (resultados) podrían ser las cuerdas que soportan esa polea. Y el problema es que nuestro sistema, con sus resultados Sí/No, sólo podría decir (una vez que hubiese conseguido pasar de la primera polea) si cierta cuerda tiene una fuerza que actúa sobre ella o no, pero no el valor de la fuerza, lo cual hace imposible una solución exacta.

Podría usted abordar el problema «utilizando» pesos exactos, como, por ejemplo, presencia o ausencia absoluta de un peso. Y luego añadir un nodo de «control» que tuviese como entradas los equilibrios en ciertos puntos, y como salidas pesos que van colocándose en un sitio u otro, dependiendo de las entradas, hasta que el sistema llega al equilibrio. Pero quizás adoleciera de falta de precisión.

Por otra parte, por supuesto, podría ser que funcionase. No hay, ciertamente, nada mejor que montar un sistema que uno cree que puede hacer algo y ver si realmente lo hace. A fin de cuentas, eso es lo que todo el mundo hace. Es lo que se conoce como Investigación.

10. UN EXPERTO BASADO EN REGLAS Y ESCRITO EN BASIC

10.1 UN SISTEMA QUE FUNCIONA HACIA ATRAS(*)

Es posible que el sistema descrito en la primera parte de este libro le resulte a usted útil, pero el capítulo anterior pone de manifiesto que hay algunos métodos alternativos, que, en ciertas circunstancias, podrían serle más útiles. En este capítulo, usted puede encontrar una alternativa práctica.

Uno de los problemas más graves en la construcción de un sistema experto es el que se presenta cuando, aún disponiendo de todo el conocimiento necesario, resulta que éste está expresado de una forma poco conveniente.

Suponga que quiere usted construir un sistema que sea experto en el campo del diagnóstico médico.

En este caso, no es en absoluto necesario construir un sistema basado en el aprendizaje a partir de ejemplos, como hacíamos para la predicción meteorológica, porque hay mucha información fácilmente disponible que permite llegar a una solución mucho más directa. El problema que se presenta es que la información no está en la forma adecuada.

(*) (N. del T.) Es cierto que el sistema que se describe en este capítulo «funciona hacia atrás», en el sentido de que, en cada momento, con la información disponible (síntomas introducidos hasta ese momento) determina la enfermedad más probable, y, si lo juzga necesario *vuelve* para pedir más información. Pero no es en absoluto un sistema con *encadenamiento hacia atrás*, tal como se definió en el capítulo anterior, por la sencilla razón de que no hace ningún encadenamiento de reglas ni de nodos.

Abra una enciclopedia médica y busque, por ejemplo, la gripe. Verá que se dan todos los síntomas, sin ambigüedad alguna. En otras palabras, dados los síntomas, siempre se puede hacer un diagnóstico preciso.

Pero, para utilizar la información de ese modo, parece que lo que habría que hacer sería coger un paciente, imaginar que tiene gripe, y consultar la enciclopedia para ver si presenta los síntomas correspondientes. Y puede intuirse que esta forma de proceder no es la más correcta.

Lo que habría que hacer sería ver a un paciente, descubrir sus síntomas, y considerar estos síntomas para ver de qué padece, y la enciclopedia no parece que sea el medio más adecuado para esta forma de proceder. No necesitamos algo que nos muestre una enfermedad con muchos síntomas, sino un sistema que, a partir de un grupo de síntomas, dé una enfermedad. Y eso es lo que vamos a construir ahora. La situación ideal sería aquella en la que usted pudiera, dado un campo concreto, entregarle a la máquina un montón de definiciones, y que ella pudiera utilizar esas definiciones de forma parecida a como lo haría un experto humano.

Desde luego, eso es lo que tratan de hacer los programas como PUFF, DENDRAL y PROSPECTOR, de modo que no hay nada de malo en intentarlo.

Utilizaremos un mecanismo de inferencia bayesiano, para tener en cuenta el hecho de que la mayoría de las informaciones no son, en absoluto, seguras, sino probabilísticas, y para tener en cuenta también el hecho de que, cuando hable usted con la gente sobre su sistema, quedará muy bien si dice que usa un mecanismo de inferencia bayesiano. Y haremos énfasis sobre todo en la forma de la información que habrá de darle usted al programa, referente al campo en el que se supone que ha de ser experto, porque la recopilación de esa información es seguramente la parte más difícil de la tarea.

Comencemos, pues, con la codificación:

```
2000 DATA SINTOMAS
2010 DATA 1, SINTOMA1
2020 DATA 2, SINTOMA2, END
```

Esa es la forma en la que guardaremos los síntomas. Al decir «síntomas», parece que estuviéramos refiriéndonos exclusivamente al campo médico, pero realmente podría ser cualquier otro. La esencia del asunto es que hay un montón de preguntas que el ordenador puede hacer,

y que esas preguntas se guardan como cadenas llamadas SINTOMA1, SINTOMA2, etc.

Por ejemplo, SINTOMA1 podría ser la cadena: ¿TOSE MUCHO EL PACIENTE?. O, si está tratando de arreglar un coche rebelde, podría ser: ¿ESTAN DEBILES LAS LUCES?

Organizando así las cosas, pueden introducirse muy cómodamente cuantas preguntas se quiera.

Luego, tenemos las enfermedades:

```
1000 DATA ENFERMEDADES
1010 DATA ENFERMEDAD1,p,(j,ps,pn,) 999
1020 DATA ENFERMEDAD2,p,(j,ps,pn,) 999
```

Esa es la forma en la que guardaremos las enfermedades. Que tampoco tienen por qué ser enfermedades. Pueden ser resultados cualesquiera: cada sentencia DATA contiene un resultado y toda la información relacionada con él.

Examinando los elementos uno a uno, el primero que aparece es el nombre del resultado (por ejemplo, gripe). El siguiente, p, es la probabilidad *a priori* de ese resultado, P(H), es decir, la probabilidad de que se dé ese resultado, supuesto que no hay ninguna información más. Luego tenemos una serie de elementos repetidos, con tres componentes cada uno. La primera, j, es el número de un síntoma aplicable (o de una variable, si se prefiere llamar así), y las dos siguientes son P(E:H) y P(E: no H) probabilidades de obtener una respuesta positiva para la presencia de esta variable, supuesto que el resultado es cierto y supuesto que el resultado es falso. El último elemento de todos es un código de parada, que sirve para que el programa sepa dónde acaban los datos relativos a una determinada enfermedad.

Por ejemplo:

```
1010   DATA GRIPE, .01, 1, .9, .01, 2, 1, .01, 3, 0, .01, 999
```

según esto, hay una probabilidad *a priori* P(H) = 0.01 de que cualquier persona elegida al azar tenga la gripe.

Supongamos ahora que el programa hace la pregunta 1 (síntoma 1). Tenemos P(E:H) = 0.9 y P(E: no H) = 0.01, lo que significa que si el paciente tiene gripe, entonces responderá «sí» a esta pregunta nueve veces de cada diez, y que, si no la tiene, sólo responderá «sí» una vez de cada cien. Evidentemente, una respuesta «sí» apoya a la hipótesis de que tiene gripe, y una «no» tiende a indicar lo contrario.

Lo mismo puede decirse del segundo grupo de síntoma/probabilidades (2, 1, 0.01). En este caso, P(E:H) = 1, lo que indica que, si tiene gripe, tiene necesariamente que presentar este síntoma. Podría presentar el síntoma sin tener gripe, pero es poco probable (P(E:no H) = 0.01).

La pregunta 3 excluye a la gripe en el caso de que la respuesta sea «sí», porque P(E:H) = 0. Ello podría corresponder a una pregunta como: «¿Viene el paciente padeciendo los mismos síntomas desde hace mucho tiempo?», o algo parecido.

El obtener cifras aceptables para estas probabilidades requiere un poco de reflexión, y, si se quieren conseguir buenos resultados, exige cierta investigación. Y, para ser francos, el obtener toda esa información es seguramente la tarea más difícil, y en la que el ordenador no puede ayudar mucho. Pero se puede escribir un programa de propósito general que permita manejar esa información, si es que se consigue llegar a tenerla.

Fundamental en tal programa es el teorema de Bayes:

$$P(H:E) = \frac{P(E:H)P(H)}{P(E:H)P(H) + P(E:\text{no } H)\ P(\text{no } H)}$$

Lo esencial es que la probabilidad de cierta hipótesis, dado cierto elemento de evidencia, puede calcularse a partir de la probabilidad *a priori* de tal hipótesis (sin saber nada de la evidencia) y de las probabilidades de que se presente la evidencia, supuesto que la hipótesis es cierta y que es falsa.

Por tanto, considerando nuestras enfermedades, podemos calcular:

$$P(H:E) = \frac{ps*p}{ps*p + pn*(1-p)}$$

El proceso consiste en empezar con P(H) = p para cada una de las enfermedades. El programa plantea una pregunta y calcula P(H:E) de acuerdo con la respuesta. La respuesta «sí» conduce al cálculo definido por la fórmula anterior; para la respuesta «no» se aplicaría la misma fórmula, pero sustituyendo ps por (1—ps), y pn por (1—pn).

Hecho esto, la pregunta queda «olvidada», pero el efecto de su respuesta ha sido el de sustituir la probabilidad *a priori* P(H), por P(H:E). Y el proceso sigue del mismo modo, actualizando continuamente las P(H) de todas las enfermedades, conforme va llegando nueva información.

En términos generales, podemos dividir el programa en las siguientes partes:

PRIMERA PARTE

El programa explora las sentencias DATA para averiguar cuántas enfermedades y cuántos síntomas hay. Podría dársele este dato previamente, pero de este modo se evita uno la molestia de tener que contarlos. Con el resultado, pueden DIMensionarse las variables que se necesiten.

SEGUNDA PARTE

El programa busca en las sentencias DATA todas las probabilidades *a priori*, y calcula los valores de regla VR(I), para ver qué preguntas (síntomas) son las más importantes y saber así qué tiene que preguntar primero. Si se calcula:

RV(I) = RV(I) + ABS(P(H:E)—P(H:no E))

para todas las preguntas, y considerando todas las enfermedades en las que interviene cada pregunta, se conseguirá que los valores de VR(I) representen la magnitud del cambio que las correspondientes preguntas pueden provocar en las probabilidades del conjunto de enfermedades a las que se aplican.

TERCERA PARTE

El programa descubre cuál es la pregunta más importante, y la plantea. La respuesta puede darse de diferentes formas: podría ser simplemente «sí» o «no»; también podría incluirse «no sé» (que no produciría ningún cambio), o, lo que es algo más sofisticado, puede darse en una escala de —5 a +5 para expresar distintos grados de certidumbre en la respuesta.

CUARTA PARTE

Dada la evidencia obtenida, se actualizan las probabilidades *a priori*.

QUINTA PARTE

Se calculan nuevos valores de reglas. Asimismo, se calculan los valo-

res máximos y mínimos para todas las enfermedades, basándose en las probabilidades *a priori* actuales y en la suposición de que toda la evidencia que falta actuaría a favor o en contra de cada hipótesis. La idea es ver si cada una de las hipótesis tiene aún posibilidades o no de ser cierta; aquéllas que no las tienen pueden descartarse, y las que tienen valores mínimos por encima de un cierto nivel, pueden anunciarse como conclusiones posibles.

El programa vuelve entonces a la tercera parte, y continúa así hasta que no tenga nada más que hacer.

10.2 EL PROGRAMA BASIC

Damos a continuación el equivalente en BASIC de los anteriores pasos, explicando con detalle la versión en Applesoft, y listando sin más comentarios la versión para el Spectrum.

```
10  REM  ENCONTRAR EL NUMERO DE ENFE
    RMEDADES Y SINTOMAS
20  READ A$,A$
30 IL = 0:RL = 0
40  READ P,J
50  READ PY,PN,S: IF S <  > 999 THEN
    50
60 IL = IL + 1
70  READ A$: IF A$ = "SINTOMAS" THEN
    90
80  GOTO 40
90  READ A$,B$: IF A$ = "FIN" THEN 1
    20
100 RL = RL + 1
110  GOTO 90
120  DIM P(IL),RV(RL),EO%(IL),MI(IL)
    ,MA(IL),IL%(IL),ER%(RL)
130  FOR J = 1 TO RL
140 ER%(J) = 1
150  NEXT J
```

Este trozo de programa explora la lista de enfermedades, buscando y contando el número de códigos de parada para obtener IL, el número de enfermedades definidas. Luego lee la lista de síntomas y los cuenta

hasta que llega a la palabra END en las DATA, obteniendo RL como número de síntomas en la lista.

Las variables que se DIMensionan son:

P(IL)	que se utiliza para guardar las probabilidades en cada momento.
RV(RL)	que se utiliza para guardar el «valor» de cada síntoma en función de la magnitud de los cambios que puede inducir en las probabilidades de las enfermedades.
EO%(IL)	que es una lista del número de síntomas aplicables a cada enfermedad y que aún están pendientes; este número se reduce en una unidad cada vez que se pregunta por un síntoma.
MI(IL)	que es el valor mínimo que puede alcanzar la probabilidad de cada enfermedad.
MA(IL)	que es el valor máximo que puede alcanzar la probabilidad de cada enfermedad.
IL%(IL)	que contiene el número de síntomas que hay en la lista para cada enfermedad.
ER%(ER)	que es un «interruptor» que se pone inicialmente con el valor 1. Después de cada pregunta se pone a 0, para evitar que se repita esa misma pregunta.

```
160  REM  ENCONTRAR LA PRIORIDAD DE
     PROBABILIDADES Y LOS VALORES REG
     ULADOS
170  RESTORE : READ A$
180  FOR I = 1 TO IL
190  READ A$,P(I),J
200 P = P(I)
210  READ PY,PN,S
220 EO%(I) = EO%(I) + 1
230 RV(J) = RV(J) +  ABS (P * PY / (
     P * PY + (1 - P) * PN) - P * (1 -
     PY) / (P * (1 - PY) + (1 - P) *
     (1 - PN)))
240  IF S <  > 999 THEN J = S: GOTO
     210
250 IL%(I) = EO%(I)
260  NEXT I
270  REM  ENCONTRAR EL SINTOMA MAXIM
     O Y LA PREGUNTA
```

```
280 R = 0:HR = 0
290  FOR J = 1 TO RL
300  IF RV(J) > R THEN HR = J:R = RV
    (J)
310 RV(J) = 0
320  NEXT J
330  IF HR = 0 THEN  PRINT "NO HAY N
    UEVOS SINTOMAS": END
340  READ A$
350  FOR J = 1 TO HR
360  READ P,A$
370  NEXT J
380  HOME : PRINT ,"EXPERTO": PRINT
    ,"--------": PRINT : PRINT : PRINT
    "PREGUNTA:": PRINT A$
390 ER%(HR) = 0
```

En este punto, el programa ha encontrado una pregunta y la ha escrito en la pantalla. Lo que el usuario tiene que hacer es responder a ella. La respuesta, en una escala de —5 a +5, se guarda en RE.

```
400  INPUT "RESPONDA EN UNA ESCALA E
    NTRE -5 (NO) Y  +5 (SI) ";RE
410  REM  OBTENER LA PROBABILIDAD PR
    IORITARIA USANDO LA RESPUESTA
420  RESTORE : READ A$
430  FOR I = 1 TO IL
440  READ A$,P
450  FOR K = 1 TO IL%(I)
460  READ J,PY,PN
470  IF J <  > HR OR EO%(I) = 0 THEN
    540
480 EO%(I) = EO%(I) - 1
490 P = P(I)
500 PE = P * PY + (1 - P) * PN
510  IF RE > 0 THEN P(I) = P * (1 +
    (PY / PE - 1) * RE / 5)
520  IF RE <  = 0 THEN P(I) = P * (1
     + (PY - (1 - PY) * PE / (1 - PE
    )) * RE / 5)
```

```
530  IF P(I) =  INT (P(I)) THEN EO%(
     I) = 0
540  NEXT K
550  READ ST
560  NEXT I
```

Quizás esté usted un poco perplejo ante los cálculos de P(I), que son diferentes según que RE sea positiva o negativa. La idea es la siguiente:

Si RE fuera +5, entonces la respuesta sería «Sí», y calcularíamos:

$$P(H:E) = \frac{P(E:H)P(H)}{P(E:H)P(H) + P(E:\text{no } H)P(\text{no } H)}$$

y si fuera —5, calcularíamos:

$$P(H:\text{no } E) = \frac{P(\text{no } E:H)P(H)}{P(\text{no } E:H)P(H) + P(\text{no } E:\text{no } H)P(\text{no } H)}$$

y si fuera 0, calcularíamos:

$$P(H:E) = P(H).$$

Y si RE tiene cualquier valor comprendido entre los extremos, pondremos un poco de una cosa y un poco de la otra. El programa que damos calcula precisamente cuánto hay que poner de una cosa y de la otra, organizando los cálculos de modo que la ejecución sea lo más rápida posible.

Si P(I) = INT(P(I)) podemos hacer EO%(I) = 0, porque ya tenemos un suceso seguro, y no hacen falta más preguntas.

```
570  REM  ENCONTRAR LOS NUEVOS VALOR
     ES REGULADOS Y LA PROBABILIDAD M
     AXIMA Y MINIMA
580  RESTORE
590  READ A$
600 MM = 0:MH = 0
610  FOR I = 1 TO IL
620 P = P(I)
630 A1 = 1:A2 = 1:A3 = 1:A4 = 1
640  READ A$,PZ
```

```
650  FOR K = 1 TO IL%(I)
660  READ J,PY,PN
670  IF ER%(J) * EO%(I) = 0 THEN 740

680  IF PN > PY THEN PY = 1 - PY:PN =
     1 - PN
690 RV(J) = RV(J) + P * PY / (P * PY
     + (1 - P) * PN) - P * (1 - PY) /
     (P * (1 - PY) + (1 - P) * (1 - P
     N))
700 A1 = A1 * PY
710 A2 = A2 * PN
720 A3 = A3 * (1 - PY)
730 A4 = A4 * (1 - PN)
740  NEXT K
750 MA(I) = P * A1 / (P * A1 + (1 -
     P) * A2)
760 MI(I) = P * A3 / (P * A3 + (1 -
     P) * A4)
770  IF MA(I) < PZ THEN EO%(I) = 0
780  IF MI(I) > MM THEN MH = I:MM =
     MI(I)
790  READ A$
800  NEXT I
```

Aquí, lo que se refiere al cálculo de los nuevos valores de regla está claro, pero quizás no se vea tan claramente lo que concierne a MI(I) y MA(I).

En primer lugar, obsérvese que comparamos PY con PN para ver cuál es el mayor. Si el mayor es PY, muy bien, eso significa que una respuesta «Sí» aumenta P(H) y una respuesta «No» la reduce. Si el mayor es PN, necesitamos que ocurra lo contrario, y eso significa, cuando se trabaja con probabilidades, tomar los complementos, es decir, 1—PY y 1—PN. De este modo, podemos expresar las preguntas como queramos, y, siempre y cuando las probabilidades sean correctas, el programa sabrá si la evidencia apoya o no a cada hipótesis.

Lo que se hace entonces es calcular MA(I), la máxima probabilidad posible, suponiendo que todas las preguntas cuya respuesta aún no se ha dado, tendrían una respuesta en favor de la hipótesis. Y este cálculo es el mismo que se hace para obtener P(H:E) cuando se están actualizando

las probabilidades, pero, para conseguir el máximo valor posible, se considera que E no es simplemente la respuesta a una pregunta, sino la respuesta a todas las preguntas pendientes. Así, por ejemplo, P(E:H) es la probabilidad de que toda la evidencia en favor de la hipótesis aparezca, suponiendo que la hipótesis es cierta. Y, suponiendo que todas las preguntas son independientes entre sí, ello es igual al producto de todos los valores pendientes de P(E:H): P(aparezca toda la evidencia a favor:H) = $P(E_1:H)P(E_2:H)P(E_3:H)...P(E_n:H)$, que es el valor que se guarda en A1.

Así, tendremos:

A1 = P(aparezca toda la evidencia a favor: H)
A2 = P(aparezca toda la evidencia a favor: no H)
A3 = P(aparezca toda la evidencia en contra: H)
A4 = P(aparezca toda la evidencia en contra: no H)

Y, francamente, es comprensible que el lector empiece a estar algo confuso. El procedimiento que yo adopto normalmente, consiste en apuntar lo que creo que significa y luego irme a cortar el césped un rato. Cuando vuelvo, escribo lo que debería haber escrito antes si lo hubiera pensado mejor. Luego me voy otra vez a cortar el césped. Y entonces me doy cuenta de que no estaba del todo en lo cierto, y vuelvo a modificar las ecuaciones. Y de nuevo a cortar el césped. Y así sucesivamente.

Caída la tarde, el césped ha quedado magnífico y yo me voy a tomar una copa, y en ese momento me doy cuenta de que me había equivocado. Justo un instante antes de dormirme, veo qué es lo que tendría que haber hecho, pero no llego a escribirlo.

Al día siguiente, el proceso se repite.

Cuando acaba la semana, descubro que tengo un suelo de piedra en el lugar donde antes había un bonito césped y que estoy agotado por la fatiga. Entonces me doy cuenta de que las primeras ecuaciones que escribí eran las correctas, y de que he perdido el papel donde estaban escritas.

También usted puede consumir así su vida. Ello tiene la gran ventaja de cualificarle para una pensión de invalidez antes de cumplir los 35 años. Pero también puede planificar sistemáticamente desde el principio todo el asunto.

Sea como sea, una vez obtenidos A1, A2, A3 y A4, usted puede calcular como se indica los valores mínimos y máximos posibles. Todo lo que tiene que hacer es copiar el programa y reflexionar sobre el hecho de que el escribir este libro ha supuesto tremendos sufrimientos.

Finalmente, deducimos la conclusión más probable:

```
810  REM  BUSCAR UN CLARO GANADOR
820  FOR I = 1 TO IL
830  IF MI(MH) <  = MA(I) AND I <  >
     MH THEN MM = 0
840  NEXT I
850  IF MM = 0 THEN 270
860  RESTORE : READ A$
870  FOR I = 1 TO MH
880  READ I$,A$
890  FOR K = 1 TO IL%(I)
900  READ J,PY,PN
910  NEXT K
920  READ A$
930  NEXT I
940  HOME : PRINT ,"EXPERTO": PRINT
     ,"--------": PRINT : PRINT : PRINT
     "EL RESULTADO MAS PROBABLE ES ";
     I$
950  PRINT "CON UNA PROBABILIDAD DE
     ";P(MH)
960  END
1000  DATA  ENFERMEDADES
1010  DATA  ENFERMEDAD 1,.01,1,.8,.1
     ,2,.2,.9,999
1020  DATA  ENFERMEDAD 2,.5,1,1,.5,2
     ,0,.1,999
2000  DATA  SINTOMAS
2010  DATA  1,SINTOMA 1
2020  DATA  2,SINTOMA 2,FIN,FIN
```

Lo que hace este trozo de programa es tomar el máximo de todos los valores mínimos, y ver si es mayor que cualquiera de los máximos de los otros valores.

Si es así, se ha acabado. Si no, será porque algún otro resultado podría llegar a ser más probable que el actual, y, como éste es, de momento, el más probable, aún no se puede sacar una conclusión y hay que volver a pedir otra información.

Si le parece que esto es demasiado rebuscado, podría, simplemente, elegir aquel resultado que tuviese la mayor P(I) de acuerdo con las informaciones dadas en cada momento; para un valor suficientemente alto de esta probabilidad, podría decidir que ésta es la conclusión correcta. O suficientemente correcta.

He aquí el programa completo, escrito en BASIC para el Spectrum:

Listado para el Sinclair Spectrum

```
   2 REM Advierta que los codigos de las s
entencias DATA y los tests correspondiente
s para parada estan alterados ligeramente
para el SPECTRUM
   5 DIM a$(20): DIM b$(20): DIM i$(20)
  10 REM Encuentra el numero de enfermedad
es y sintomas
  20 READ a$,a$
  30 LET i1=0: LET r1=0
  40 READ p,j
  50 READ py,pn,s: IF s<>999 THEN  GO TO 5
0
  60 LET i1=i1+1
  70 READ a$: IF a$(  TO 8)="sintomas" THE
N  GO TO 90
  80 GO TO 40
  90 READ p,a$: IF a$(  TO 3)="fin" THEN
GO TO 120
 100 LET r1=r1+1
 110  GO TO 90
 120 DIM p(i1): DIM v(r1): DIM o(i1): DIM
i(i1): DIM a(i1): DIM l(i1): DIM r(r1)
 130 FOR j=1 TO r1
 140 LET r(j)=1
 150 NEXT j
 160 REM Encuentra la probabilidad a prior
i y los valores de regla
 170 RESTORE : READ a$
 180 FOR i=1 TO i1
 190 READ a$,p(i),j
 200 LET p=p(i)
 210 READ py,pn,s
```

```
 220 LET o(i)=o(i)+1
 230 LET v(j)=v(j)+ABS (p*py/(p*py+(1-p)*p
n)-p*(1-py)/(p*(1-py)+(1-p)*(1-pn)))
 240 IF s<>999 THEN  LET j=s: GO TO 210
 250 LET l(i)=o(i)
 260 NEXT i
 270 REM encuentra el sintoma mas importan
te y pregunta
 280 LET r=0: LET hr=0
 290 FOR j=1 TO rl
 300 IF v(j)>r THEN  LET hr=j: LET r=v(j)
 310 LET v(j)=0
 320 NEXT j
 330 IF hr=0 THEN  PRINT "No mas sintomas"
: STOP
 340 READ a$
 350 FOR j=1 TO hr
 360 READ p,a$
 370 NEXT j
 380 CLS : PRINT AT 12,0;,"EXPERTO"',"____
___"''"Pregunta:"'(a$)
 390 LET r(hr)=0
 400 INPUT "Responde dentro de una escala
de -5 a 5: ";re
 410 REM Actualiza las probabilidades a pr
iori
 420 RESTORE : READ a$
 430 FOR i=1 TO il
 440 READ a$,p
 450 FOR k=1 TO l(i)
 460 READ j,py,pn
 470 IF j<>hr OR o(i)=0 THEN  GO TO 540
 480 LET o(i)=o(i)-1
 490 LET p=p(i)
 500 LET pe=p*py+(1-p)*pn
 510 IF re>0 THEN  LET p(i)=p*(1+(py/pe-1)
*re/5)
 520 IF re<=0 THEN  LET p(i)=p*(1+(py-(1-p
y)*pe/(1-pe))*re/5)
 530 IF p(i)=INT (p(i)) THEN  LET o(i)=0
 540 NEXT k
```

```
 550 READ st
 560 NEXT i
 570 REM encuentra nuevos valores de regla
, maximos y minimos
 580 RESTORE
 590 READ a$
 600 LET mm=0: LET mh=0
 610 FOR i=1 TO il
 620 LET p=p(i)
 630 LET a1=1: LET a2=1: LET a3=1: LET a4=
1
 640 READ a$,pz
 650 FOR k=1 TO l(i)
 660 READ j,py,pn
 670 IF r(j)*o(i)=0 THEN  GO TO 740
 680 IF pn>py THEN  LET py=1-py: LET pn=1-
pn
 690 LET v(j)=v(j)+p*py/(p*py+(1-p)*pn)-p*
(1-py)/(p*(1-py)+(1-p)*(1-pn))
 700 LET a1=a1*py
 710 LET a2=a2*pn
 720 LET a3=a3*(1-py)
 730 LET a4=a4*(1-pn)
 740 NEXT k
 750 LET a(i)=p*a1/(p*a1+(1-p)*a2)
 760 LET i(i)=p*a3/(p*a3+(1-p)*a4)
 770 IF a(i)<pz THEN  LET o(i)=0
 780 IF i(i)>mm THEN  LET mh=i: LET mm=i(i
)
 790 READ s
 800 NEXT i
 810 REM busca una hipotesis claramente fa
vorita
 820 FOR i=1 TO il
 830 IF i(mh)<=a(i) AND i<>mh THEN  LET mm
=0
 840 NEXT i
 850 IF mm10 THEN  GO TO 270
 860  RESTORE : READ a$
 870 FOR i=1 TO mh
 880 READ i$,p
```

```
890 FOR k=1 TO l(i)
900 READ j,py,pn
910 NEXT k
 920 READ s
 930 NEXT i
 940 CLS : PRINT AT 12,0;,"EXPERTO"',"____
___"''"El resultado mas probable es "'(i$)
''"Con probabilidad ";(p(mh))
 960 STOP
1000 DATA "Enfermedades"
1010  DATA "Enfermedad 1",.01,1,.8,.1,2,.2
,.9,999
1020 DATA "Enfermedad 2",.5,1,1,.5,2,0,.1,
999
2000 DATA "Sintomas"
2010 DATA 1,"Sintoma 1"
2020 DATA 2,"Sintoma 2",0,"Fin"
```

Aunque este programa funciona razonablemente bien (un milagro de la tecnología moderna, verdaderamente), sus prestaciones dependen en gran medida de la lista de preguntas que incluya usted.

A nivel teórico, todos los cálculos se realizan bajo la suposición de que todas las preguntas son independientes entre sí. Y, si no ocurre así, el funcionamiento no será tan bueno.

Si, por ejemplo, se pregunta: «¿tiene usted la temperatura alta?», y «¿tiene usted fiebre?», es evidente que las respuestas están muy correlacionadas. No tiene ningún sentido plantear ambas preguntas, pero, si se hace tal cosa, el resultado será que se alterarán las probabilidades asociadas con esos síntomas.

Por otra parte, hay que proporcionar suficientes preguntas para que el sistema pueda realmente llegar a una conclusión, es decir, hacer algún tipo de diagnóstico. Esto parece obvio, pero suponga que le da usted detalles del Resfriado común y de la Gripe, y que los síntomas consisten en sentirse con malestar general, con mucosidad nasal y con tos. Quizás las probabilidades sean algo diferentes en cada caso, pero han de serlo lo suficiente como para permitir que el programa vea realmente la diferencia. El truco está en buscar preguntas claras y no ambiguas, que no se solapen unas con otras y que sean capaces de establecer clasificaciones netas.

Normalmente, acabará usted trabajando igual que lo han hecho las personas que ya han desarrollado sistemas expertos:

Escribirá el programa y lo acompañará de un puñado de reglas que le parecen adecuadas. (Las reglas son esas definiciones que se dan en las sentencias DATA).

Entonces, haciendo como que tiene una neumonía (si se trata de un sistema médico; en el caso de un experto en mecánica podría ser una batería agotada) responderá a las preguntas que le hará el sistema.

El sistema generará una respuesta falsa o planteará alguna pregunta que no viene a cuento, y usted empezará a jugar con las preguntas y las probabilidades hasta que el funcionamiento mejore. Y, en realidad, este proceso no tiene gran cosa que ver con los ordenadores, sino con la comprensión de la materia en la que quiere usted que el ordenador se haga experto, y puede ser en sí mismo bastante interesante. Y, si a usted no le interesa, siempre puede volver al sistema descrito en la primera parte del libro, echarle algunos ejemplos y dejarle que trabaje por su cuenta.

10.3 UNA BASE DE CONOCIMIENTOS MÉDICOS

Si introduce usted en su ordenador el esquema de inferencia bayesiana que acabamos de describir, para que pueda hacer algo útil le faltará todavía una cosa: una Base de Conocimientos. Ahora bien, estrictamente hablando, es cosa suya proporcionar el conocimiento específico sobre algún tema. Pero el hacerlo puede ser bastante laborioso. Por tanto, para ayudarle un poco, a continuación puede encontrar una Base de Conocimientos para el tema del Diagnóstico Médico. Si carga usted esas sentencias a partir de las líneas 1000 y 2000, el experto tendrá conocimientos sobre cerca de 100 enfermedades diferentes y sus diagnósticos. Los números que se dan son bastante precisos, y, si bien todo ello no le va a cualificar para practicar la Medicina, sí que le permitirá entregarse a ciertas fantasías hipocondríacas. También servirá para darle una idea sobre cómo podría usted construir una Base de Conocimientos en otros campos, y le mostrará cómo funciona el sistema experto cuando lleva incluido un bloque real de conocimientos.

Una vez que lo haya cargado, pruebe a modificar alguno de los elementos para ver cómo ello afecta a la habilidad del experto.

Por ejemplo, usted puede hacer que el experto tenga en cuenta que la Bronquitis Crónica (enfermedad 10) es más común entre los hombres

que entre las mujeres («síntoma» 53). Para ello, puede añadirle a la enfermedad 10:

53, 0.8, 0.5,

es decir, si el paciente tiene bronquitis crónica, la probabilidad de que sea hombre es 0.8, y, si no la tiene, la probabilidad de que sea hombre es 0.5. Dicho de otro modo, en la población sana hay tantos hombres como mujeres.

La incidencia de muchas enfermedades depende del sexo, por lo que la pregunta 53 puede añadirse en bastantes puntos de la Base de Conocimientos. Si lo hace usted así, se encontrará con que el valor de RV(53) aumenta, y una de las primeras cosas que pregunta el experto al comenzar una consulta, será el sexo del paciente. Lo cual parece razonable: pocos médicos (humanos) se avendrían a dar un diagnóstico sin este elemento básico de conocimiento.

Luego pruebe a incluir otra enfermedad, como la peste bubónica, por ejemplo, y vea si el experto puede diagnosticarla. Si necesita añadir más síntomas (preguntas), puede hacerlo fácilmente, al final de la lista, referenciándolos por su número, lo mismo que se hace con los ya existentes.

Ensaye finalmente en escribir su propia Base de Conocimientos, con preguntas y «enfermedades» relativas, por ejemplo, al problema de por qué no arranca su coche (probablemente habrá contraído la peste bubónica).

```
1000 DATA ENFERMEDADES

1010 DATA RESFRIADO COMUN , 0.02 ,1 ,0.9 ,
     0.05 , 2 ,0.8 ,0.02 ,3 ,0.8 ,0.02 ,
     5 ,0.6 ,0.01 ,6 ,1 ,0.01 ,7 ,0.2 ,
     0.01 ,8 ,0.5 ,0.01 ,15 ,0.8 ,0.01 ,
     34 ,0 ,0.01 ,999
1020 DATA RINITIS ALERGICA , 0.01 ,1 ,1 ,
     0.01 ,2 ,1 ,0.01 ,6 ,0.9 ,0.01 ,
     10 ,0.7 ,0.01 ,11 ,0.7 ,0.01 ,12 ,
     0.6 ,0.01 ,20 ,0.9 ,0.01 ,999
1030 DATA SINUSITIS ,0.01 ,14 ,0.8 ,0.01 ,
     13 ,0.9 ,0.01 ,15 ,0.8 ,0.01 ,
     7 ,0.8 ,0.01 ,22 ,0.5 ,0.01 ,2 ,0.5 ,
```

```
     0.01 ,6 ,0.5 ,0.01 ,
     63 ,0.9 ,0.01 ,999
1040 DATA FARINGITIS , 0.02 ,3 ,1 ,0.01 ,
     16 ,0.9 ,0.01 ,8 ,0.5 ,0.01 ,
     11 ,0.9 ,0.01 ,37 ,0.8 ,0.3 ,64 ,
     0.4 ,0.01 ,999
1050 DATA AMIGDALITIS ,0.001 ,3 ,1 ,0.01 ,
     7 ,0.9 ,0.01 ,15 ,1 ,0.01 ,16 ,0.7 ,
     0.01 ,19 ,0 ,0.5 ,8 ,0.9 ,0.01 ,
     34 ,0 ,0.01 ,64 ,0.8 ,0.01 ,999
1060 DATA GRIPE ,0.01 ,3 ,0.9 ,0.01 ,
     1 ,0.9 ,0.01 ,6 ,0.5 ,0.01 ,
     7 ,0.7 ,0.01 ,8 ,1 ,0.01 ,
     15 ,1 ,0.01 ,17 ,0.8 ,0.01 ,
     18 ,0.6 ,0.01 ,34 ,0 ,0.01 ,999
1070 DATA LARINGITIS ,0.01 ,4 ,1 ,0.01 ,
     8 ,0.6 ,0.01 ,15 ,0.05 ,0.01 ,
     16 ,0.7 ,0.01 ,37 ,0.8 ,0.3 ,
     5 ,0.9 ,0.01 ,21 ,0.1 ,0.01 ,999
1080 DATA TUMOR DE LARINGE ,0.00004 ,
     4 ,1 ,0.01 ,34 ,0.99 ,0.01 ,
     37 ,0.8 ,0.3 ,999
1090 DATA BRONQUITIS AGUDA ,0.005 ,
     5 ,1 ,0.01 ,8 ,1 ,0.01 ,12 ,1 ,0.01 ,
     15 ,1 ,0.01 ,18 ,0.5 ,0.01 ,
     21 ,1 ,0.01 ,31 ,0.9 ,0.01 ,
     34 ,0 ,0.01 ,22 ,0.9 ,0.01 ,999
1100 DATA BRONQUITIS CRONICA ,0.005 ,
     5 ,1 ,0.01 ,12 ,0.9 ,0.01 ,
     14 ,0.5 ,0.01 ,21 ,1 ,0.01 ,
     22 ,0.8 ,0.01 ,34 ,1 ,0.01 ,
     36 ,0.9 ,0.01 ,37 ,0.8 ,0.3 ,999
1110 DATA ASMA ,0.02 ,12 ,0.8 ,0.01 ,
     22 ,1 ,0.01 ,23 ,0.5 ,0.01 ,
     24 ,0.5 ,0.01 ,25 ,0.5 ,0.01 ,
     26 ,0.5 ,0.01 ,31 ,0.8 ,0.01 ,999
1120 DATA ENFISEMA PULMONAR ,0.01 ,
     22 ,1 ,0.01 ,5 ,0.6 ,0.01 ,
     26 ,0.8 ,0.01 ,12 ,0.6 ,0.01 ,
     21 ,0.6 ,0.01 ,37 ,0.8 ,0.3 ,999
1130 DATA NEUMONIA ,0.003 ,8 ,1 ,0.01 ,
```

```
     15 ,1 ,0.01 ,18 ,0.8 ,0.01 ,
     22 ,1 ,0.01 ,23 ,0.5 ,0.01 ,
     26 ,0.5 ,0.01 ,28 ,0.1 ,0.01 ,
     29 ,0.02 ,0.01 ,27 ,0.2 ,0.01 ,
     31 ,0.9 ,0.01 ,36 ,0.1 ,0.9 ,
     7 ,0.9 ,0.01 ,17 ,0.9 ,0.01 ,
     32 ,0.5 ,0.005 ,999
1140 DATA PLEURESIA ,0.001 ,
     31 ,0.8 ,0.01 ,32 ,0.8 ,0.01 ,
     22 ,0.5 ,0.01 ,5 ,0.8 ,0.01 ,
     8 ,0.9 ,0.01 ,15 ,1 ,0.01 ,
     34 ,0 ,0.01 ,999
1150 DATA NEUMOTORAX ,0.0002 ,
     18 ,0.8 ,0.01 ,22 ,0.8 ,0.01 ,
     32 ,0.8 ,0.005 ,999
1160 DATA BRONQUIECTASIA , 1E-5 ,
     21 ,1 ,0.01 ,27 ,0.5 ,0.01 ,
     5 ,1 ,0.01 ,14 ,0.5 ,0.01 ,999
1170 DATA ABCESO PULMONAR ,1E-5 ,
     33 ,0.9 ,0.01 ,18 ,0.5 ,0.01 ,
     21 ,0.5 ,0.01 ,27 ,0.5 ,0.01 ,999
1180 DATA NEUMOCONIOSIS ,0.001 ,
     22 ,1 ,0.01 ,36 ,1 ,0.01 ,
     21 ,0.8 ,0.01 ,9 ,1 ,0.01 ,999
1190 DATA CANCER DE PULMON ,0.001 ,
     5 ,1 ,0.01 ,21 ,0.8 ,0.01 ,
     27 ,0.5 ,0.01 ,22 ,0.5 ,0.01 ,
     18 ,0.8 ,0.01 ,12 ,0.5 ,0.01 ,
     37 ,0.99 ,0.3 ,999
1200 DATA FIBROSIS INTERSTICIAL ,1E-5 ,
     22 ,0.8 ,0.01 ,35 ,0.8 ,0.01 ,
     21 ,0.6 ,0.01 ,999
1210 DATA EDEMA PULMONAR ,0.001 ,
     22 ,0.9 ,0.01 ,25 ,0.9 ,0.01 ,
     30 ,0.5 ,0.01 ,27 ,0.5 ,0.01 ,
     26 ,0.5 ,0.01 ,12 ,0.8 ,0.01 ,999
1220 DATA GASTRITIS ,0.01 ,41 ,0.8 ,0.01 ,
     43 ,0.8 ,0.01 ,42 ,0.5 ,0.01 ,
     8 ,0.4 ,0.01 ,37 ,0.9 ,0.5 ,999
1230 DATA HERNIA DEL HIATO ,0.001 ,
     18 ,0.9 ,0.01 ,32 ,0.5 ,0.005 ,
```

```
     42 ,0.8 ,0.005 ,57 ,0.9 ,0.01 ,
     16 ,0.9 ,0.01 ,41 ,0.8 ,0.01 ,999
1240 DATA ULCERA DUODENAL ,0.01 ,
     37 ,0.8 ,0.2 ,42 ,0.99 ,0.005 ,
     41 ,0.8 ,0.01 ,57 ,0.7 ,0.01 ,999
1250 DATA ULCERA GASTRICA ,0.01 ,
     42 ,0.9 ,0.005 ,46 ,0.5 ,0.01 ,
     20 ,0.8 ,0.01 ,41 ,0.7 ,0.01 ,
     56 ,0.9 ,0.01 ,57 ,0.7 ,0.01 ,
     62 ,0.0001 ,0.01 ,999
1260 DATA DIVERTICULITIS INTESTINAL ,
     0.001 ,42 ,0.6 ,0.005 ,
     43 ,0.5 ,0.01 ,41 ,0.5 ,0.01 ,
     8 ,0.5 ,0.01 ,56 ,0.8 ,0.01 ,999
1270 DATA ENTEROCOLITIS ,0.0001 ,
     42 ,0.9 ,0.005 ,43 ,0.7 ,0.01 ,
     23 ,0.3 ,0.01 ,41 ,0.3 ,0.01 ,
     8 ,0.3 ,0.01 ,999
1280 DATA ENFERMEDAD DE CROHN ,0.0001 ,
     42 ,0.9 ,0.005 ,43 ,0.9 ,0.01 ,
     15 ,0.9 ,0.01 ,8 ,0.7 ,0.01 ,
     62 ,0.00001 ,0.01 ,999
1290 DATA OCLUSION INTESTINAL ,0.00001 ,
     42 ,0.9 ,0.005 ,43 ,0.8 ,0.01 ,
     41 ,0.5 ,0.01 ,999
1300 DATA APENDICITIS ,0.001 ,
     34 ,0.1 ,0.9 ,42 ,0.9 ,0.005 ,
     41 ,0.8 ,0.01 ,
     8 ,0.8 ,0.01 ,44 ,0 ,0.5 ,999
1310 DATA INTOXICACION ALIMENTARIA ,
     0.001 ,42 ,0.5 ,0.005 ,
     41 ,0.9 ,0.01 ,
     43 ,0.9 ,0.01 ,7 ,0.8 ,0.01 ,999
1320 DATA GASTROENTERITIS ,0.01 ,
     41 ,0.8 ,0.01 ,42 ,0.7 ,0.005 ,
     43 ,0.9 ,0.01 ,8 ,0.5 ,0.01 ,999
1330 DATA LITIASIS RENAL ,0.001 ,
     42 ,0.7 ,0.005 ,999
1340 DATA PIELONEFRITIS AGUDA ,0.001 ,
     42 ,0.9 ,0.005 ,8 ,0.8 ,0.01 ,
     41 ,0.7 ,0.01 ,67 ,0.9 ,0.01 ,999
```

```
1350 DATA LITIASIS BILIAR ,0.01 ,
     42 ,0.5 ,0.005 ,
     41 ,0.5 ,0.01 ,57 ,0.9 ,0.01 ,999
1360 DATA COLECISTITIS ,0.001 ,
     42 ,0.8 ,0.005 ,8 ,0.9 ,0.01 ,
     41 ,0.8 ,0.01 ,45 ,0.8 ,0.001 ,999
1370 DATA HERPES ,0.001 ,14 ,0.8 ,0.01 ,
     18 ,0.5 ,0.01 ,60 ,0.9 ,0.01 ,
     59 ,0.9 ,0.01 ,2 ,0.6 ,0.01 ,
     8 ,0.5 ,0.01 ,999
1380 DATA TROMBOFLEBITIS ,0.0005 ,
     40 ,0.8 ,0.01 ,999
1390 DATA ARTRITIS REUMATOIDE ,0.001 ,
     15 ,0.8 ,0.01 ,17 ,0.8 ,0.01 ,
     40 ,0.5 ,0.001 ,999
1400 DATA INSUFICIENCIA CARDIACA ,0.001 ,
     22 ,0.9 ,0.01 ,36 ,0.5 ,0.01 ,
     25 ,0.5 ,0.001 ,12 ,0.6 ,0.01 ,
     18 ,0.5 ,0.01 ,32 ,0.3 ,0.001 ,
     39 ,0.5 ,0.01 ,42 ,0.5 ,0.01 ,
     26 ,0.3 ,0.001 ,47 ,0.9 ,0.01 ,999
1410 DATA ANSIEDAD ,0.01 ,46 ,0.9 ,0.01 ,
     28 ,0.3 ,0.01 ,47 ,0.6 ,0.01 ,
     39 ,0.8 ,0.01 ,23 ,0.6 ,0.01 ,
     48 ,0.6 ,0.01 ,16 ,0.3 ,0.01 ,
     43 ,0.2 ,0.01 ,22 ,0.5 ,0.01 ,
     50 ,0.5 ,0.01 ,57 ,0.5 ,0.01 ,
     58 ,0.5 ,0.01 ,15 ,0.5 ,0.01 ,
     7 ,0.5 ,0.01 ,38 ,0.5 ,0.01 ,999
1420 DATA DEPRESION ,0.01 ,47 ,0.5 ,0.01 ,
     7 ,0.5 ,0.01 ,49 ,0.5 ,0.01 ,
     50 ,0.5 ,0.01 ,15 ,0.5 ,0.01 ,
     62 ,0.8 ,0.01 ,999
1430 DATA TROMBOSIS CORONARIA ,0.01 ,
     18 ,0.5 ,0.01 ,32 ,0.9 ,0.001 ,
     20 ,0.5 ,0.01 ,36 ,0 ,0.2 ,
     38 ,0.5 ,0.01 ,22 ,0.5 ,0.01 ,
     23 ,0.5 ,0.01 ,
     41 ,0.5 ,0.01 ,15 ,0.9 ,0.01 ,999
1440 DATA ANGINA DE PECHO ,0.01 ,
     37 ,0.8 ,0.3 ,18 ,0.9 ,0.01 ,
```

```
36 ,0.9 ,0.01 ,22 ,0.5 ,0.01 ,
23 ,0.5 ,0.01 ,38 ,0.5 ,0.01 ,
20 ,0.9 ,0.01 ,32 ,1 ,0.001 ,999
1450 DATA EMBOLIA PULMONAR ,0.0001 ,
     22 ,1 ,0.01 ,18 ,0.7 ,0.01 ,
     21 ,0.6 ,0.01 ,27 ,0.5 ,0.001 ,
     25 ,0.5 ,0.001 ,26 ,0.4 ,0.001 ,999
1460 DATA APOPLEGIA ,0.001 ,
     28 ,0.8 ,0.01 ,38 ,0.7 ,0.01 ,
     51 ,0.8 ,0.001 ,58 ,0.9 ,0.01 ,
     61 ,0.9 ,0.01 ,999
1470 DATA ATAQUE ISQUEMICO TRANSITORIO ,
     0.001 ,28 ,0.8 ,0.01 ,38 ,0.7 ,0.01 ,
     51 ,0.8 ,0.001 ,34 ,0 ,0.01 ,
     20 ,0.5 ,0.01 ,58 ,0.9 ,0.01 ,
     61 ,0.9 ,0.01 ,999
1480 DATA TUBERCULOSIS ,0.0001 ,
     7 ,0.5 ,0.01 ,8 ,0.5 ,0.01 ,
     12 ,0.5 ,0.01 ,15 ,0.5 ,0.01 ,
     18 ,0.5 ,0.01 ,5 ,0.5 ,0.01 ,
     30 ,0.5 ,0.01 ,27 ,0.5 ,0.001 ,
     22 ,0.5 ,0.01 ,62 ,0.0001 ,0.01 ,
     23 ,0.5 ,0.01 ,999
1490 DATA HEMORROIDES ,0.01 ,
     52 ,0.9 ,0.001 ,49 ,0.8 ,0.01 ,
     56 ,0.9 ,0.01 ,59 ,0.5 ,0.01 ,999
1500 DATA HIPOTIROIDISMO ,0.001 ,
     49 ,0.8 ,0.01 ,17 ,0.5 ,0.01 ,
     24 ,0 ,0.01 ,23 ,0.001 ,0.01 ,
     39 ,0.001 ,0.01 ,4 ,0.5 ,0.01 ,
     43 ,0 ,0.01 ,46 ,0.001 ,0.01 ,
     48 ,0.001 ,0.01 ,62 ,0.9 ,0.05 ,999
1510 DATA COLON IRRITABLE ,0.0007 ,
     43 ,0.5 ,0.01 ,49 ,0.5 ,0.01 ,
     42 ,0.8 ,0.001 ,41 ,0.3 ,0.01 ,
     57 ,0.9 ,0.01 ,56 ,0.5 ,0.01 ,999
1520 DATA CANCER INTESTINAL ,0.001 ,
     43 ,0.9 ,0.01 ,49 ,0.9 ,0.01 ,
     52 ,0.5 ,0.001 ,42 ,0.5 ,0.005 ,
     56 ,0.9 ,0.01 ,62 ,0.0001 ,0.01 ,999
```

```
1530 DATA COLITIS ULCEROSA ,0.0004 ,
     42 ,0.8 ,0.005 ,43 ,0.8 ,0.01 ,
     52 ,0.6 ,0.001 , 23 ,0.5 ,0.01 ,
     41 ,0.5 ,0.01 ,8 ,0.5 ,0.01 ,
     34 ,0.4 ,0.01 , 56 ,0.9 ,0.01 ,999
1540 DATA ENFERMEDAD DE MENIERE ,0.0005 ,
     38 ,0.9 ,0.001 ,41 ,0.8 ,0.01 ,
     34 ,0.5 ,0.01 ,20 ,0.9 ,0.01 ,999
1550 DATA ESPONDILOSIS CERVICAL ,0.006 ,
     54 ,0.9 ,0.01 ,7 ,0.5 ,0.01 ,
     38 ,0.5 ,0.01 ,58 ,0.9 ,0.01 ,
     61 ,0.5 ,0.01 ,999
1560 DATA HEMORRAGIA SUBDURAL ,1E-6 ,
     55 ,0.99 ,0.001 ,28 ,0.9 ,0.001 ,
     28 ,0.9 ,0.01 ,7 ,0.9 ,0.01 ,
     41 ,0.9 ,0.01 ,38 ,0.9 ,0.01 ,
     20 ,0.5 ,0.01 ,34 ,0.5 ,0.01 ,999
1570 DATA TUMOR CEREBRAL ,1E-6 ,
     7 ,0.9 ,0.01 ,41 ,0.9 ,0.01 ,
     38 ,0.8 ,0.01 ,50 ,0.8 ,0.01 ,
     34 ,0.5 ,0.01 ,999
1580 DATA MENINGITIS ,1E-6 ,8 ,0.9 ,0.01 ,
     7 ,0.9 ,0.01 ,41 ,0.9 ,0.01 ,
     28 ,0.7 ,0.01 ,54 ,0.9 ,0.01 ,
     2 ,0.9 ,0.01 ,60 ,0.5 ,0.01 ,999
1590 DATA HEMORRAGIA SUBARACNOIDEA ,1E-5 ,
     7 ,0.99 ,0.01 ,54 ,0.9 ,0.01 ,
     38 ,0.7 ,0.01 ,28 ,0.7 ,0.01 ,
     41 ,0.8 ,0.01 ,2 ,0.8 ,0.01 ,999
1600 DATA GLAUCOMA AGUDO ,0.01 ,
     2 ,0.9 ,0.01 ,7 ,0.9 ,0.01 ,
     41 ,0.7 ,0.01 ,20 ,0.8 ,0.01 ,
     34 ,0.8 ,0.01 ,63 ,0.8 ,0.01 ,
     68 ,0.9 ,0.01 ,999
1610 DATA ARTERIRIS DE LA TEMPORAL ,
     0.001 ,7 ,0.9 ,0.01 ,8 ,0.7 ,0.01 ,
     17 ,0.7 ,0.01 ,23 ,0.8 ,0.01 ,
     63 ,0.99 ,0.01 ,999
1620 DATA DISPEPSIA ,0.1 ,7 ,0.7 ,0.01 ,
     57 ,0.7 ,0.01 ,42 ,0.7 ,0.01 ,
     41 ,0.7 ,0.01 ,62 ,0.001 ,0.01 ,
```

```
     20 ,0.9 ,0.01 ,999
1630 DATA BLOQUEO CARDIACO ,0.0003 ,
     22 ,0.5 ,0.01 ,58 ,0.8 ,0.01 ,
     39 ,0.6 ,0.01 ,18 ,0.6 ,0.01 ,999
1640 DATA ANEMIA PERNICIOSA ,0.0004 ,
     22 ,0.9 ,0.01 ,58 ,0.9 ,0.01 ,
     39 ,0.9 ,0.01 ,36 ,0.9 ,0.01 ,
     45 ,0.5 ,0.01 ,42 ,0.5 ,0.01 ,
     50 ,0.5 ,0.01 ,28 ,0.4 ,0.01 ,999
1650 DATA JAQUECA ,0.1 ,7 ,1 ,0.01 ,
     15 ,0.9 ,0.01 ,41 ,0.9 ,0.01 ,
     43 ,0.5 ,0.01 ,2 ,0.9 ,0.01 ,
     20 ,0.9 ,0.01 ,34 ,0.9 ,0.01 ,
     63 ,0.99 ,0.01 ,999
1660 DATA HIPERTENSION ESENCIAL ,0.15 ,
     7 ,0.5 ,0.01 ,39 ,0.5 ,0.01 ,
     15 ,0.9 ,0.01 ,34 ,0.9 ,0.01 ,999
1670 DATA ECCEMA ,0.03 ,59 ,0.9 ,0.01 ,
     60 ,1 ,0.01 ,999
1680 DATA URTICARIA ,0.03 ,59 ,0.9 ,0.01 ,
     60 ,1 ,0.01 ,46 ,0.5 ,0.01 ,999
1690 DATA SARNA ,0.001 ,59 ,1 ,0.01 ,
     60 ,1 ,0.01 ,999
1700 DATA SARAMPION ,0.02 ,15 ,1 ,0.01 ,
     8 ,1 ,0.01 ,6 ,0.9 ,0.01 ,
     2 ,0.9 ,0.01 ,11 ,0.9 ,0.01 ,
     5 ,0.9 ,0.01 ,43 ,0.5 ,0.01 ,
     60 ,1 ,0.01 ,7 ,0.5 ,0.01 ,
     34 ,0 ,0.01 ,999
1710 DATA RUBEOLA ,0.01 ,8 ,0.5 ,0.01 ,
     60 ,0.9 ,0.01 ,54 ,0.2 ,0.01 ,
     34 ,0 ,0.01 ,64 ,0.5 ,0.01 ,999
1720 DATA VARICELA ,0.001 ,60 ,1 ,0.01 ,
     59 ,1 ,0.01 ,8 ,0.8 ,0.01 ,
     7 ,0.5 ,0.01 ,15 ,0.5 ,0.01 ,
     34 ,0 ,0.01 ,999
1730 DATA PSORIASIS ,0.02 ,46 ,0.6 ,0.01 ,
     3 ,0.5 ,0.01 ,60 ,0.99 ,0.01 ,
     59 ,0.5 ,0.01 ,999
1740 DATA PITIRIASIS RUBRA ,0.01 ,60 ,
     1 ,0.01 ,59 ,0.9 ,0.01 ,
```

```
     34 ,0.5 ,0.01 ,999
1750 DATA ACNE ROSACEO ,0.01 ,
     60 ,0.9 ,0.01 ,2 ,0.5 ,0.01 ,
     34 ,0.8 ,0.01 ,999
1760 DATA TIROTOXICOSIS ,0.001 ,
     46 ,0.9 ,0.01 ,47 ,0.8 ,0.01 ,
     48 ,0.9 ,0.01 ,23 ,0.9 ,0.01 ,
     39 ,0.9 ,0.01 ,22 ,0.8 ,0.01 ,
     43 ,0.8 ,0.01 ,62 ,0.00001 ,0.01 ,
     2 ,0.5 ,0.01 ,24 ,0.9 ,0.01 ,
     64 ,0.3 ,0.01 ,68 ,0.3 ,0.01 ,999
1770 DATA DIABETES MELLITUS ,0.01 ,
     62 ,0.0001 ,0.01 ,61 ,0.5 ,0.01 ,
     2 ,0.5 ,0.01 ,66 ,0.99 ,0.01 ,
     68 ,1 ,0.01 ,999
1780 DATA CANCER DE ESTOMAGO ,0.0003 ,
     41 ,0.5 ,0.01 ,42 ,0.7 ,0.005 ,
     62 ,0.0001 ,0.01 ,52 ,0.6 ,0.001 ,
     56 ,0.5 ,0.01 ,999
1790 DATA FIBRILACION AURICULAR ,0.001 ,
     39 ,0.8 ,0.01 ,38 ,0.5 ,0.01 ,
     20 ,0.4 ,0.01 ,58 ,0.5 ,0.01 ,999
1800 DATA ENFERMEDAD DE HODGKIN ,0.0001 ,
     23 ,0.5 ,0.01 ,63 ,0.6 ,0.01 ,
     54 ,0.8 ,0.01 ,59 ,0.7 ,0.01 ,
     64 ,0.99 ,0.01 ,999
1810 DATA FIEBRE GLANDULAR ,0.001 ,
     8 ,0.9 ,0.01 ,7 ,0.9 ,0.01 ,
     3 ,0.9 ,0.01 ,15 ,0.9 ,0.01 ,
     64 ,0.8 ,0.01 ,54 ,0.8 ,0.01 ,
     45 ,0.5 ,0.001 ,60 ,0.5 ,0.01 ,999
1820 DATA LINFOMA ,0.0001 ,64 ,0.9 ,0.01 ,
     54 ,0.8 ,0.01 ,15 ,0.8 ,0.01 ,
     62 ,0.001 ,0.01 ,8 ,0.8 ,0.01 ,
     23 ,0.5 ,0.01 ,59 ,0.8 ,0.01 ,999
1830 DATA PAPERAS ,0.01 ,64 ,0.99 ,0.01 ,
     8 ,0.8 ,0.01 ,15 ,0.9 ,0.01 ,
     16 ,0.7 ,0.01 ,54 ,0.6 ,0.01 ,
     3 ,0.8 ,0.01 ,999
1840 DATA PARALISIS FACIAL ,0.0003 ,
     51 ,0.9 ,0.01 ,63 ,0.5 ,0.01 ,
```

```
     2 ,0.9 ,0.01 ,999
1850 DATA ENFERMEDAD DE PARKINSON ,0.001 ,
     48 ,0.9 ,0.01 ,51 ,0.8 ,0.01 ,
     53 ,0.8 ,0.5 ,50 ,0.2 ,0.01 ,
     28 ,0.2 ,0.01 ,34 , 1 ,0.01 ,999
1860 DATA ARTRITIS REUMATOIDE ,0.01 ,
     7 ,0.8 ,0.01 ,15 ,0.8 ,0.01 ,
     8 ,0.8 ,0.01 ,64 ,0.8 ,0.01 ,
     60 ,0.5 ,0.01 ,59 ,0.001 ,0.01 ,
     48 ,0.1 ,0.01 ,999
1870 DATA CISTITIS ,0.01 ,66 ,0.9 ,0.01 ,
     65 ,0.9 ,0.01 ,67 ,0.9 ,0.01 ,
     8 ,0.5 ,0.01 ,999
1880 DATA TUMOR RENAL ,0.001 ,
     8 ,0.6 ,0.01 ,62 ,0.0001 ,0.01 ,
     41 ,0.5 ,0.01 ,42 ,0.5 ,0.01 ,
     65 ,0.7 ,0.01 ,999
1890 DATA TUMOR EN LA VEJIGA ,0.0004 ,
     65 ,0.9 ,0.01 ,42 ,0.5 ,0.01 ,
     66 ,0.5 ,0.01 ,67 ,0.5 ,0.01 ,
     8 ,0.3 ,0.01 ,999
1900 DATA IRITIS ,0.0005 ,2 ,0.9 ,0.01 ,
     68 ,0.9 ,0.01 ,999
1910 DATA HEPATITIS AGUDA ,0.001 ,
     8 ,0.8 ,0.01 ,15 ,0.8 ,0.01 ,
     17 ,0.5 ,0.01 ,42 ,0.5 ,0.01 ,
     45 ,0.5 ,0.01 ,41 ,0.5 ,0.01 ,999
2000 DATA SINTOMAS
2010 DATA  1, ESTORNUDAS MUCHO?
2020 DATA  2, TE DUELEN LOS OJOS, O
     LAGRIMEAS MUCHO?
2030 DATA  3, TE DUELE LA GARGANTA?
2040 DATA  4, TIENES RONQUERA O AFONIA?
2050 DATA  5, TOSES MUCHO?
2060 DATA  6, TIENES MUCOSIDAD NASAL
     MUY FLUIDA?
2070 DATA  7, TE DUELE LA CABEZA, O
     SUFRES, EN GENERAL, DE DOLORES
     DE CABEZA?
2080 DATA  8, TIENES ALTA LA TEMPERATURA?
     (SUPERIOR A 37 GR.)
```

```
2090 DATA  9, TE VES OBLIGADO, POR TU
     TRABAJO, A PASAR MUCHAS HORAS
     RESPIRANDO AIRE VICIADO
     O POLVORIENTO?
2100 DATA 10, TE PICA LA NARIZ?
2110 DATA 11, TIENES LA GARGANTA RESECA?
2120 DATA 12, RESPIRAS CON DIFICULTAD
     Y FATIGA?
2130 DATA 13, TIENES LA NARIZ MUY
     TAPONADA?
2140 DATA 14, HAS TENIDO RECIENTEMENTE UN
     RESFRIADO U OTRA INFECCION SIMILAR?
2150 DATA 15, TIENES MALESTAR GENERAL?
2160 DATA 16, TIENES DIFICULTAD PARA
     TRAGAR?
2170 DATA 17, TIENES DOLORES MUSCULARES?
2180 DATA 18, SIENTES CUALQUIER TIPO DE
     DOLOR EN EL PECHO?
2190 DATA 19, TE HAN EXTIRPADO LAS
     AMIGDALAS?
2200 DATA 20, TUS SINTOMAS TIENDEN A
     APARECER EN "ATAQUES", Y NO A ESTAR
     PRESENTES CONTINUAMENTE?
2210 DATA 21, TIENES UNA TOS PRODUCTIVA,
     O SEA, UNA TOS EN LA QUE
     EXPULSAS ALGO?
2220 DATA 22, RESPIRAS ENTRECORTADAMENTE?
2230 DATA 23, SUDAS MUCHO? (NO YA CUANDO
     HACES ALGUN ESFUERZO, SINO CUANDO
     ESTAS FISICAMENTE RELAJADO)
2240 DATA 24, TIENES LA FRECUENCIA
     CARDIACA ELEVADA? (NORMALMENTE,
     DEBE ESTAR ENTRE 60 Y 80 PULSACIONES
     POR MINUTO, Y ALGO MAS RAPIDA PARA
     PERSONAS DE EDAD SUPERIOR A 70
     O INFERIOR A 20)
2250 DATA 25, SUFRES ATAQUES DE
     DIFICULTADES RESPIRATORIAS TAN
     FUERTES QUE LLEGAN A PREOCUPARTE?
2260 DATA 26, TIENES CIANOSIS, ES DECIR,
     SE TE HA PUESTO LA PIEL
```

```
     DE UN COLOR LIGERAMENTE AZULADO?
2270 DATA 27, CUANDO TOSES,
     PRESENTAN SANGRE LOS ESPUTOS?
2280 DATA 28, ESTAS ATURDIDO O PERPLEJO
     SOBRE CUANTO ACONTECE A TU ALREDEDOR?
2290 DATA 29, ESTAS (O ESTA EL PACIENTE)
     EN ESTADO DELIRANTE, HABLANDO
     INCOHERENTEMENTE Y CON MALA
     COORDINACION MUSCULAR?
2300 DATA 30, TIENES UNA TOS SECA
     (NO PRODUCTIVA)?
2310 DATA 31, SIENTES DOLOR AL RESPIRAR
     O AL TOSER?
2320 DATA 32, HAS SENTIDO EN ESTOS DIAS
     UN DOLOR MUY FUERTE EN EL PECHO?
2330 DATA 33, TIENES SENSACIONES
     ALTERNANTES DE FRIO Y DE CALOR?
2340 DATA 34, HACE YA TIEMPO
     (SEIS SEMANAS O MAS) QUE PRESENTAS
     ALGUNO DE ESOS SINTOMAS?
2350 DATA 35, TIENES LOS DEDOS
     HIPOCRATICOS, O EN "PALILLOS DE
     TAMBOR"? (SE LLAMAN ASI CUANDO LAS
     CUTICULAS CASI HAN DESAPARECIDO,
     Y LAS UNAS SE ABOMBAN EXCESIVAMENTE
     HACIA LA PUNTA)
2360 DATA 36, TIENES SINTOMAS QUE
     APARECEN FUNDAMENTALMENTE AL
     REALIZAR ALGUN ESFUERZO?
2370 DATA 37, FUMAS?. RESPONDE CON EL
     NUMERO QUE RESULTA DE DIVIDIR POR
     CINCO EL NUMERO DE CIGARRILLOS
     DIARIOS (ASI, SI FUMAS 20, HAS DE
     RESPONDER 4). RECUERDA QUE EL NUMERO
     MAXIMO PARA LA RESPUESTA ES 5.
     SI NO FUMAS, RESPONDE -5.
2380 DATA 38, TIENES SENSACIONES DE
     VERTIGO?
2390 DATA 39, TIENES PALPITACIONES?
     (SENSACION DE QUE EL CORAZON LATE
     MAS FUERTE, O MAS RAPIDO, O MENOS
```

```
     REGULARMENTE DE LO QUE DEBIERA)
2400 DATA 40, TIENES ALGUN TOBILLO
     EXCESIVAMENTE HINCHADO?
2410 DATA 41, TIENES VOMITOS,
     O FUERTES NAUSEAS?
2420 DATA 42, SUFRES DE ALGUN TIPO DE
     DOLOR ABDOMINAL, O DOLOR LOCALIZADO
     ENTRE LAS COSTILLAS INFERIORES
     Y LAS INGLES?
2430 DATA 43, PADECES DIARREA, O SUELES
     TENER CRISIS DE DIARREA?
2440 DATA 44, TE HAN EXTIRPADO EL
     APENDICE?
2450 DATA 45, TIENES ICTERICIA?
     (LA ICTERICIA NO ES UNA ENFERMEDAD,
     SINO UN SINTOMA. CON FRECUENCIA
     ES MAS PATENTE EN LOS OJOS:
     LA PARTE BLANCA SE TORNA AMARILLENTA)
2460 DATA 46, TE SIENTES TENSO O INQUIETO?
2470 DATA 47, TE CUESTA CONCILIAR EL
     SUENO, O TE DESVELAS FRECUENTEMENTE
     POR LA NOCHE?
2480 DATA 48, TIENES CONTRACCIONES
     INVOLUNTARIAS O TEMBLORES?
2490 DATA 49, SUFRES DE ESTRENIMIENTO, O
     SUELES TENER CRISIS DE ESTRENIMIENTO?
2500 DATA 50, TE FALLA LA MEMORIA?.
     TIENES DIFICULTAD PARA RECORDAR
     HECHOS AISLADOS, YA SEA OCASIONAL
     O REGULARMENTE?
2510 DATA 51, SUFRES AFASIA, ES DECIR,
     HAS PERDIDO CAPACIDAD PARA EXPRESARTE
     CON PALABRAS?
2520 DATA 52, HAS TENIDO ALGUNA HEMORRAGIA
     POR EL RECTO?
2530 DATA 53, ERES HOMBRE O MUJER?.
     RESPONDE CON 5 PARA HOMBRE Y -5 PARA
     MUJER. UNA RESPUESTA 0 HACE QUE EL
     DIAGNOSTICO SEA
     INDEPENDIENTE DEL SEXO
2540 DATA 54, TIENES RIGIDEZ Y/O
```

```
     DOLOR EN EL CUELLO?
2550 DATA 55, HAS SUFRIDO ALGUN GOLPE O
     HERIDA DE CUALQUIER TIPO EN LA CABEZA
     DURANTE LAS ULTIMAS SEMANAS?
     (UN GOLPE INCLUSO LEVE PUEDE
     SER IMPORTANTE)
2560 DATA 56, HAS EXPULSADO RECIENTEMENTE
     HECES DE ASPECTO ANORMAL?
2570 DATA 57, EXPULSAS MUCHOS GASES,
     POR CUALQUIER VIA?
2580 DATA 58, TIENES SENSACIONES
     REPENTINAS DE DESMAYO, ES DECIR,
     DE SENTIRTE DESFALLECIDO Y MAREADO,
     INCLUSO, QUIZAS, CON PERDIDA
     DE CONSCIENCIA?

2590 DATA 59, TE PICA ALGUNA PARTE DEL
     CUERPO, TENGAS O NO ERUPCION EN ELLA?
2600 DATA 60, TIENES ALGUNA ERUPCION O
     LESION DE CUALQUIER TIPO EN LA PIEL?
2610 DATA 61, TIENES ALGUNA PARTE DEL
     CUERPO ENTUMECIDA, O CON SENSACION
     DE COMEZON U HORMIGUEO, COMO DE
     'AGUJAS Y ALFILERES'?
2620 DATA 62, ES TU PESO ANORMAL?
     RESPONDE CON 5 PARA UN PESO
     EXCESIVAMENTE ELEVADO,
     CON -5 SI ES EXCESIVAMENTE BAJO
     Y 0 SI ES NORMAL
2630 DATA 63, SUFRES DE ALGUN DOLOR EN
     LA CARA O EN LA FRENTE?
2640 DATA 64, TIENES ALGUN BULTO,
     HINCHAZON O ABULTAMIENTO
     EN CUALQUIER PARTE DEL CUERPO?
2650 DATA 65, ES ANORMAL EL COLOR
     DE TU ORINA?
2660 DATA 66, ORINAS CON UNA
     FRECUENCIA ANORMAL?
2670 DATA 67, TIENES DOLORES AL ORINAR?
2680 DATA 68, TIENES ALGUNA ANORMALIDAD
     EN LA VISION, COMO VER BORROSO
```

```
O DOBLE, O LUCES DESTELLANTES?
(NO CUENTAN LOS DEFECTOS QUE PUEDEN
CORREGIRSE CON GAFAS)
```

11. LA TORRE DE BABEL

Sucedió hace mucho tiempo. Un grupo de informáticos, sentados alrededor de una mesa, meditaban, absortos. Y pensando, pensando, dieron en recontar cuántos transistores tenían entre todos ellos.

—¿Os dáis cuenta —dijo uno— que si juntásemos *todos* esos transistores conseguiríamos un ordenador que podría hacer *cualquier cosa*?—. Todos los demás, entre tragos de cerveza, se mostraron de acuerdo. Y poco después, aquella misma tarde, fueron a por sus soldadores y se pusieron a trabajar.

Pasó el tiempo, y la máquina que construían se hacía más y más grande, y llegó a ser evidente que, en verdad, podría realizar cualquier tarea que se le encomendase. Jamás se había construido un ordenador tan potente, y los científicos soñaban con el día en que pudieran conectarlo, y, en cuestión de segundos, encontrar respuestas completas a cualquier problema de Todo el Universo Conocido.

Conforme la máquina se iba elevando hacia el cielo, se contrataban grandes legiones de programadores para escribir programas que pudieran controlar al nuevo monstruo, y a todos ellos se les iniciaba en los ritos secretos de la programación en el lenguaje de la máquina, porque, hasta ese momento, nadie había pensado en otro lenguaje de programación.

El día de la gran conexión se acercaba, y la promesa del nuevo gran ordenador se extendía por doquier, causando no poco temor entre las gentes, que reflexionaban sobre la tremenda fuerza cognoscitiva que pronto se desencadenaría.

Y, la noche anterior a la terminación de la máquina, el mismo grupo de informáticos volvió a sentarse alrededor de la mesa, trasegando cerveza y debatiendo qué programa tendría el honor de ser el primero en ejecutarse en la nueva máquina.

—Yo creo —dijo uno sagazmente— que realmente no importa cuál ha de ser el primer programa —a lo que los otros asintieron con prudencia, inclinando sus cabezas—. Lo importante, verdaderamente, es el lenguaje en el que esté escrito.

En ese momento, los otros siguieron asintiendo con la cabeza, pero comenzaron a preguntarse qué otros lenguajes, que no fueran el de la máquina, podría haber.

Pero el que hablaba se mantuvo firme y señaló que el lenguaje de la máquina era difícil de escribir y difícil de comprender. Que estaba orientado a la máquina y no a los problemas que se pretendía resolver con ella. Y que tenía el mayor de todos los defectos: el de ser académicamente insatisfactorio.

Así, aquella misma noche, cada uno de los informáticos volvió a su casa y escribió una definición de un lenguaje de alto nivel, acompañada de un compilador. Y cuando, brillante y diáfano, amaneció el día siguiente, se volvieron a reunir a los pies de la gran máquina y decidieron que, antes de conectarla, deberían establecer, de una vez para siempre, cuál era el mejor de los lenguajes, y adoptarlo. Y, como caballeros que eran, sometieron el asunto a discusión.

Por lo que se sabe, aún están discutiendo el asunto, y la máquina jamás llegó a conectarse, y el verdadero problema fue que ninguno de ellos llegó nunca a comprender los lenguajes desarrollados por los otros.

Y, aunque este relato no sea absolutamente cierto, hay mucho de verdad en él.

Porque, cuando el único lenguaje era el lenguaje de máquina (o un ensamblador, que es algo más asequible para entender, pero que sigue siendo, esencialmente, la misma cosa), todo informático comprendía, más o menos, lo que otra persona cercana al ordenador hacía. Ahora hay tantos lenguajes de programación disponibles que es casi imposible familiarizarse con todos ellos; aunque las ideas sólo estén esbozadas, si lo están en un lenguaje poco familiar es difícil comprenderlas. El lenguaje llega a ser un obstáculo más que una ayuda.

El motivo por el que los ejemplos de este libro están en BASIC, es que éste es el lenguaje que conoce casi todo el mundo. Pero la pregunta es: ¿es el BASIC el mejor lenguaje que puede utilizarse?, ¿no sería mejor probar con otro?.

Intuitivamente, podríamos pensar que hay fuertes argumentos para utilizar, por ejemplo, el LISP, porque es el lenguaje en el que se han

escrito la mayoría de los sistemas expertos«profesionales». O, quizás, el PROLOG, un derivado del LISP (*).

Para abordar esta cuestión hemos de considerar qué es lo que puede y lo que no puede hacerse con un lenguaje, y el primer punto a destacar es que, realmente, no es posible hacer nada que no pueda hacerse con un lenguaje de máquina. Aunque para la mayoría de los lectores resulte obvio, vale la pena insistir en que todo programa escrito en un lenguaje siempre se compila antes de su ejecución (**), y que son las instrucciones de la máquina las que le dicen a ésta lo que tiene que hacer. Si una sentencia del lenguaje no puede reducirse a instrucciones de máquina, entonces no puede ejecutarse en esa máquina. Por tanto, con el lenguaje de la máquina es posible hacer todo aquello de lo que la máquina es capaz, y con cualquier otro lenguaje es imposible hacer nada más.

En consecuencia, si con un lenguaje no puede hacerse más de lo que puede hacerse con un lenguaje de máquina, ¿puede ocurrir que ese lenguaje sea tal que impida al programador hacer todo lo que sería posible con un lenguaje de máquina? Es decir, ¿puede un lenguaje reducir el rango de actividades posibles?. La respuesta es: sí. Consideremos, por ejemplo, el BASIC, y supongamos que no hubiera facilidades como PEEK y POKE. Supongamos también que, por cualquier motivo, queremos mirar en el byte 3096 de la RAM, ver lo que hay y cambiar uno de sus bits. Salvo que se usen las más extrañas manipulaciones imaginables, tal cosa es imposible, porque el BASIC no nos permite (sin usar PEEK y POKE) acceder a una dirección absoluta ni «jugar» con bits.

De hecho, este ejemplo no es especialmente raro. Los lenguajes de alto nivel no suelen ofrecer facilidades para el direccionamiento absoluto de la memoria, y, sin embargo, hay gran cantidad de aplicaciones en las que el direccionamiento absoluto es esencial. La programación en tiempo real nos proporciona multitud de ejemplos: si el ordenador tiene que leer, por ejemplo, una medida procedente de un sensor externo, entonces la programación tendrá que hacerse casi siempre en base a direcciones absolutas para introducir el dato leído en la memoria.

(*) (N. del T.) Esta frase puede molestar a los entusiastas de PROLOG, que aseguran que es el primer lenguaje rigurosamente fundamentado sobre la lógica formal y que lo ven como el lenguaje del futuro para los sistemas basados en conocimiento; en esta última tesis se han visto apoyados por la decisión de los japoneses de adoptarlo como lenguaje de base para sus proyectos de «quinta generación». El lector interesado puede encontrar un artículo de introducción al PROLOG escrito por su creador, Alain Colmerauer, y titulado «Prolog, lenguaje de la inteligencia artificial», en el número 41 (octubre 1984) de la revista «Mundo científico».

(**) (N. del T.) O se interpreta en el curso de su ejecución, como suele hacerse con los programas escritos en BASIC.

Así, en cualquier lenguaje no pueden hacerse más cosas que con un lenguaje de máquina, y, normalmente, muchas menos. Y, si es así, ¿para qué sirven los otros lenguajes?.

Dos observaciones importantes: la primera, que los otros lenguajes son más cómodos que los de máquina; la segunda, que pueden servir (aunque no siempre) como ayuda al pensamiento.

El primer punto es evidente. El BASIC es más fácil que cualquier lenguaje de máquina, tanto para aprenderlo como para usarlo. Y esto es cierto para la mayoría de los lenguajes de alto nivel.

Pero el aspecto de ayuda al pensamiento no es tan evidente, y, aunque lo fuese, puede más bien resultar ser un cuchillo de doble filo. Siguiendo con el BASIC, consideremos el bucle FOR. Es, ciertamente, una ayuda al pensamiento. Puede usted evocarlo en su mente, contemplarlo dibujado sobre un papel, escribirlo en la pantalla, y en cualquier caso está bastante claro lo que hace. Así que puede usted olvidarse de él y concentrarse en lo que hay en su interior. Y en ese momento comienza a parecer el segundo filo del cuchillo, porque hemos presupuesto que, en BASIC, estamos definitivamente condenados al uso de bucles FOR. De hecho el bucle FOR es tan fuerte ayuda al pensamiento, que, cuando nos enfrentamos a un problema de programación, la tentación es ensayar cómo disponer los bucles FOR para resolverlo, en lugar de preguntarse si tales bucles son o no, realmente, una buena forma de proceder en ese caso. Y, si no lo fueran, lo más probable es que, de todas formas, los usáramos.

Para aclarar más este punto, consideremos la dicotomía existente entre el ordenador y el problema que el ordenador tiene que resolver. El lenguaje de la máquina está orientado exclusivamente al ordenador. Si se trata simplemente de hacer que la máquina funcione, entonces el lenguaje de la máquina es el medio más eficaz para conseguir un programa corto y rápido.

Pero, en mayor o menor grado, los lenguajes de alto nivel están orientados al problema: ellos se ocupan de la máquina y le permiten a usted concentrarse en la resolución del problema. Y, con el BASIC, por ejemplo, si usted quiere trabajar sobre una matriz de números, los bucles FOR le proporcionan un medio ideal para acceder a los elementos de la matriz. El error a evitar, sin embargo, es el de pensar que, como un lenguaje está orientado al problema, está entonces orientado hacia cualquier problema que queramos resolver. Hay diferentes tipos de problemas, que pueden resolverse de diversos modos, como revelan los nombres de algunos de los lenguajes de programación. COBOL, por ejemplo, son siglas correspondientes a «Common Business Oriented Language»

(lenguaje común para actividades comerciales): está orientado a problemas comerciales. ALGOL viene de «ALGOrithmic Language» (lenguaje algorítmico): es un lenguaje indicado para problemas en los que es conveniente una solución algorítmica. (Un algoritmo es como una receta para resolver problemas, fundamentalmente en los campos de la lógica y la matemática.) FORTRAN procede de «FORmula TRANslator» (traductor de fórmulas): se diseñó para evaluar fórmulas matemáticas. ¿Y BASIC? Su significado es: «Beginners All-purpose Simbolic Instruction Code» (código de instrucciones simbólicas de propósito general para principiantes): ¡se trata de un lenguaje ideado, en realidad, para los novicios que en absoluto saben programar bien!

Pero volvamos a los sistemas expertos y a la pregunta: ¿existe un lenguaje adecuado para ellos?.

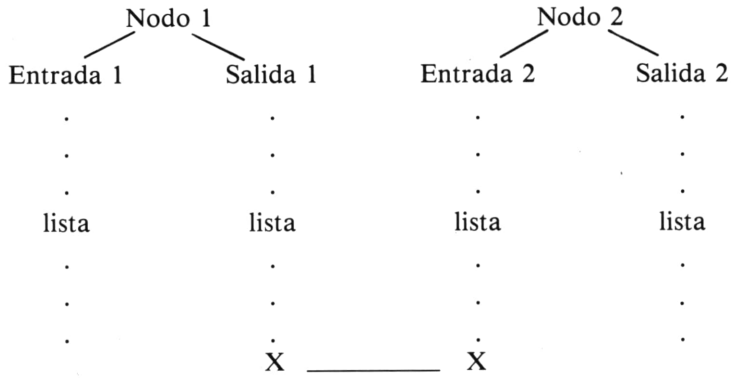
Para responder a esa pregunta, necesitamos, obviamente, conocer el tipo de problemas que surge cuando se construye un sistema experto. Y especificar tal cosa es difícil, debido, sencillamente, a la gran variedad de problemas que se subsumen bajo el encabezamiento de «sistemas expertos».

No obstante, pueden hacerse algunas observaciones. Para empezar, recordemos aquel sistema experto multinodo que habíamos ideado. Utilizando BASIC, los nodos se conectaban entre sí etiquetándolos con una variable de tipo cadena y explorando todas las variables para buscar los emparejamientos entre cadenas, que son los que determinan las conexiones. Ahora bien, esto es bastante lioso, y no ayuda precisamente el hecho de que tales variables son datos que probablemente se perderán en el momento en que se desconecte la máquina (o cuando falle la alimentación).

Este es el tipo de problema para el que se diseñó LISP, porque lo que tenemos es una serie de listas (obsérvese que LISP viene de «LISt Processing language»: lenguaje de procesamiento de listas).

Supongamos que tenemos, por ejemplo, dos nodos, nodo 1 y nodo 2, con entradas y salidas en cada uno. Definamos una lista, Nodo 2. Esta lista contendrá todas las entradas y salidas de ese nodo. Definamos dos sublistas en el Nodo 2, Entrada y Salida, para explicitar quién es quién. Hagamos lo mismo con el Nodo 1. Si entonces resulta que el Nodo 1 tiene una Salida 1 que contiene X, y que ésta está contenida en Nodo 2, Entrada 2, ese hecho puede definirse fácilmente en un programa LISP.

Podemos definir una estructura como ésta:



Y toda esta estructura puede estar contenida en el programa y no en los datos (aunque, para ser francos, hay que decir que en LISP no hay una distinción muy fuerte entre programa y datos).

Teniendo definida esta estructura, resulta especialmente fácil escribir un programa para preguntar si algunas de las entradas del Nodo 2 son salidas del Nodo 1. Ciertamente, éste es un ejemplo muy sencillo. Pero la utilidad de LISP es que nos permite definir estructuras de ese tipo, tan complejas como queramos. Podemos seguir y seguir haciendo cosas más y más complicadas, algo que, si se tratara de hacer en BASIC, podría llegar a perjudicar a un cerebro normal.

Consideremos otro aspecto. Los sistemas que hemos programado siguen un encadenamiento hacia adelante: coleccionan datos y trabajan hacia adelante a lo largo del programa, para ver qué puede hacerse con ellos. Supongamos que queremos hacer un programa con encadenamiento hacia atrás. Utilizando el ejemplo anterior, la diferencia sería la siguiente:

Con encabezamiento hacia adelante, le daríamos al programa, por ejemplo, la Entrada 1 en el Nodo 2, lo que produciría una salida a pasar al Nodo 2 como entrada, de tal modo que el Nodo 2 podría entonces producir una salida.

Con encadenamiento hacia atrás, el programa considera el Nodo 2, observa que puede producir una determinada salida, mira hacia atrás para ver cuántas entradas necesita, ve que se requiere el Nodo 1 para alguna de esas entradas, y entonces retrocede a ese Nodo 1 y pide una entrada para él. No resulta cómodo hacer todo esto en BASIC. La prin-

cipal razón de la dificultad es que BASIC no ofrece facilidades para hacer recursión (capacidad de un procedimiento para llamarse a sí mismo).

Por si no sabe usted exactamente de qué va la cosa, el ejemplo más conocido de recursión es la evaluación de un factorial (por ejemplo, $3! = 3*2*1 = 6$). Si tenemos el procedimiento recursivo FAC(N), podemos decir que:

FAC(N) = N*FAC(N—1) si N es mayor que 1
= 1 si N es menor o igual que 1.

Y así, por ejemplo, tendremos:

FAC(3) = 3*FAC(2) = 3*2*FAC(1) = 3*2*1 = 6

No siempre es tan fácil dar ejemplos cuando las situaciones son menos sencillas, pero supongamos que tuviéramos un procedimiento recursivo NODO(N). Este procedimiento evalúa el Nodo N, y lo hace considerando sus entradas, y al hacerlo también comprueba si algunas de ellas son salidas del nodo N—1; si lo son, llama a NODO(N—1), que, a su vez, puede llamar a NODO(N—2), y así hasta retroceder al comienzo de la cadena.

Pues bien, todo esto puede ser muy útil. Escribir NODO(N) puede que requiera algún esfuerzo, pero, una vez que se ha hecho puede utilizarse con las estructuras arbitrariamente complejas que mencionábamos más arriba. Naturalmente, el hecho de que la mayoría de las versiones de BASIC no incluyan la recursión, no descarta totalmente la escritura de programas recursivos: pueden hacerse en BASIC, pero es un disparate. El uso del LISP (o PROLOG) facilita las cosas cuando la recursión parece una buena idea para resolver el problema planteado.

En cierto sentido, las ventajas de esos lenguajes radican principalmente en el hecho de que están diseñados para ayudar a tratar con estructuras muy complicadas, estructuras que, en general, tienen poca relación con las matemáticas ordinarias. Y su aplicación a los sistemas expertos radica en el hecho de que, cuando intentamos que nuestro sistema experto se comporte como un experto humano, con frecuencia nos vemos abocados a hacerlo trabajar con estructuras no numéricas y bastante complejas.

El asunto de las estructuras no numéricas o no matemáticas sirve para mostrar otra cara de la moneda: la programación de fórmulas en esos lenguajes puede ser bastante laboriosa. Tomemos, por ejemplo, la sentencia de BASIC $X = 2 + 2$, y consideremos la equivalente en LISP, que

es (SETQ X ADD(2 2)). No es que sea tremendamente complicada, pero imagine lo que podría resultar para una fórmula más larga.

Lo que ocurre es que el LISP se diseñó para ayudar a resolver un tipo particular de problemas, que no es el asociado con las expresiones matemáticas. Pero algunos usuarios del LISP podrían querer aplicar algo de matemáticas, por lo que se incluyó ese parte, expresándola del mismo modo que el resto, con el resultado de que las expresiones matemáticas hay que escribirlas de una forma algo estrambótica. Con el BASIC ocurre lo contrario. Normalmente, los principiantes quieren resolver, sobre todo, problemas matemáticos; por tanto, se diseñó un lenguaje adaptado a tales problemas. Y, naturalmente, resulta que el lenguaje es algo débil en otras áreas.

Lo que necesitaríamos, realmente, sería un único y gran lenguaje que pudiera hacer todo en todas las circunstancias. Pero ello es imposible, sencillamente, porque nadie sabe como podría ser tal lenguaje. Incluso nuestros propios lenguajes naturales no sirven para todo; buena prueba de ello es el gran número de sublenguajes especializados que han ido apareciendo a lo largo del tiempo para extenderlos a nuevos campos, como, por ejemplo, el matemático, o el médico. Si, por ejemplo, quisiéramos un lenguaje que fuera perfecto para el campo de la medicina, necesitaríamos una parte de lenguaje natural para comunicar con los pacientes, una parte de lenguaje médico para comunicar con los médicos (quizás serviría el latín), y una parte de lenguaje matemático para efectuar cálculos. Y no creemos que nadie se ofenda si decimos que todo esto es más de lo que muchos médicos en ejercicio dominan.

En cierto modo, el problema de los lenguajes no es realmente tan grave como podría parecer, por la sencilla razón de que muchos de los usuarios de ordenadores sólo tienen acceso a un número limitado de lenguajes, y esto acaba con la mayoría de los argumentos sobre qué lenguaje hay que utilizar. También está el hecho de que, a menudo, el mejor lenguaje a utilizar es aquel que uno conoce bien. Por ejemplo, gran parte de la investigaciones en inteligencia artificial se han basado en el LISP, pero una razón para ello es que normalmente a los estudiantes de inteligencia artificial se les enseña el LISP, y el hábito es tan fuerte como puede ser el del BASIC en otras personas.

Pero, ciertamente, el lenguaje que uno usa, influye, de un modo u otro, en la forma de pensar en los problemas, y es muy probable que el programador de BASIC realice un sistema experto de una forma bastante diferente a como lo haría un programador de LISP.

Al fin y al cabo, el valor de un enfoque radica, simplemente, en la calidad del producto final, y, en el caso extremo, usted siempre puede descender al lenguaje de máquina. El único peligro es que se haga viejo antes de terminar de escribir el programa.

12. RESUMEN Y PANORÁMICA TÉCNICA

Dado que este libro es como una especie de historia cuya trama requiere acción por su parte, puede ser que en este momento se encuentre usted sentado y diciendo algo así:

—Muy interesante, ya entiendo cómo funcionan estos sistemas. Puedo incluso escribir mi propio sistema experto. Pero ya no me acuerdo cómo este tal Naylor ha definido las «reglas», o qué era...

Bien, podría usted acudir al índice, o consultar las notas que haya ido tomando. Pero, para hacerlo verdaderamente fácil, he aquí una lista de los términos importantes que debe usted conocer:

12.1 SUCESOS

Un suceso puede ser casi cualquier cosa.

Llamaremos H al suceso consistente en que cierta hipótesis es verdadera, y E al suceso consistente en la presencia de una determinada evidencia, que puede ir a favor o en contra de la hipótesis.

12.2 PROBABILIDADES

(véanse también las páginas 33-39)

P(H) es la probabilidad de que H sea cierta.

P(E) es la probabilidad de que E ocurra.

Las probabilidades son números comprendidos entre 0 y 1.

Si la probabilidad es 0, el suceso ocurre siempre.

P(no H) = 1—P(H) es la probabilidad de que H sea falsa.

Dos sucesos son independientes si la probabilidad de que ocurran ambos conjuntamente es igual al producto de las probabilidades de que ocurra cada uno de ellos separadamente.

$P(E_1 \text{ y } E_2)$ es la probabilidad de que ocurran E_1 y E_2 conjuntamente. E_1 y E_2 son independientes si y sólo si $P(E_1 \text{y } E_2) = P(E_1)P(E_2)$.

P(H:E) es la probabilidad de que ocurra H, supuesto que E ha ocurrido ya. Si H y E son independientes, P(H:E) = P(H). En general,

$$P(H:E) = \frac{P(H \text{ y } E)}{P(E)}$$

Análogamente,

$$P(E:H) = \frac{P(E \text{ y } H)}{P(H)}$$

De donde:

$$P(H:E) = \frac{P(E:H)P(H)}{P(E)}$$

12.2.1 Teorema de Bayes

(véanse también las páginas 47, 227-234, 242, 244)

De todo lo anterior puede deducirse el Teorema de Bayes:

$$P(H:E) = \frac{P(E:H)P(H)}{P(E:H)P(H) + P(E:\text{no } H)P(\text{no } H)}$$

Comparación entre P(H:E) y P(E:H):

De entrada, podría parecer mucho más razonable trabajar siempre con P(H:E); al fin y al cabo, lo que queremos saber es la probabilidad de cada hipótesis, dados los elementos de evidencia disponibles, no a la inversa.

El problema es que P(H:E) no es en absoluto una cantidad conocida.

Si así fuese, no tendría sentido escribir un programa de ordenador para obtenerla.

Por otro lado, normalmente es mucho más fácil conocer las P(E:H) cuando se recogen los datos para el problema a resolver.

Planteamos las preguntas: si H es cierta, ¿cuál es la probabilidad de obtener este elemento de evidencia concreto?. Y, para P(E:no H) si H es falsa, ¿cuál es la probabilidad de observar este elemento de evidencia concreto?

Las respuestas a esas dos preguntas permiten al programa calcular P(H:E), mejorando sucesivamente la estimación, conforme se van añadiendo elementos de evidencia.

Para dos (o más) sucesos, E_1 y E_2, si estos sucesos son independientes, tendremos:

$$P(E_1 \text{ y } E_2{:}H) = P(H{:}E_1)P(H{:}E_2)$$

Si E es el suceso «todos los E_i ocurren», y los E_i son independientes entre sí, entonces se puede calcular:

$$P(E{:}H) = P(E_1{:}H)P(E_2{:}H) \ldots P(E_n{:}H)$$
$$P(\text{no } E{:}H) = P(\text{no } E_1{:}H)P(\text{no } E_2{:}H) \ldots P(\text{no } E_n{:}H)$$

12.2.2 **Probabilidades** a priori **y** a posteriori

Supongamos que tenemos una hipótesis H y alguna evidencia, a favor o en contra de H, a la que llamaremos E.

Entonces, P(H) es la probabilidad *a priori* de H: probabilidad de que H sea cierta, sin tener ningún conocimiento sobre E. P(H:E), o P(H:no E), es la probabilidad *a posteriori* de H: probabilidad de que H sea cierta, una vez que conocemos la verdad acerca de E.

Por el Teorema de Bayes, tendremos:

$$P(H{:}E) = \frac{P(E{:}H)P(H)}{P(E{:}H)P(H) + P(E{:}\text{no } H)P(\text{no } H)}$$

o bien, si es que E no ocurre:

$$P(H{:}\text{no}E) = \frac{P(\text{no } E{:}H)P(H)}{P(\text{no}E{:}H)P(H) + P(\text{no } E{:}\text{no } H)P(\text{no } H)}$$

Supongamos que, dada una H, hay un gran número de elementos de evidencia, progresivamente disponibles, que sirven bien para apoyar H o para contradecirla; llamémosles E_1, E_2, ..., E_n.

Si dispusiéramos de entrada de todos ellos, y si todos fueran independientes entre sí, podríamos calcular P(E:H) como el producto de las $P(E_i:H)$, y luego calcular P(H:E), donde E es el suceso «todos los E_i ocurren». Análogamente, podríamos calcular P(no E:H) como el producto de todas las P(no E_i:H).

Pero puede ser más conveniente trabajar por etapas, acumulando la evidencia y sus efectos conforme vamos adquiriendo los E_i. Podemos, para ello, utilizar las probabilidades *a priori* y *a posteriori* del siguiente modo:

1. P(H) es la probabilidad *a priori* de H.
2. Dado un elemento de evidencia, E_i, tendremos $P(E_i:H)$ y $P(E_i$:no H).
3. Utilizando el Teorema de Bayes, podemos calcular $P(H:E_i)$ o P(H:no E_i), dependiendo del resultado de E_i; ésta es la probabilidad *a posteriori* de H.
4. Ahora, podemos olvidarnos por completo de E_i, y considerar a la probabilidad *a posteriori* de H como la nueva probabilidad *a priori* de H. Así, hacemos $P(H) = P(H:E_i)$, o bien P(H) = P(H:no E_i), dependiendo del resultado de E_i.
5. Seleccionamos un nuevo E_i a considerar, y volvemos al paso 1.

12.2.3 Posibilidades

La posibilidad a favor o en contra de un suceso puede calcularse a partir de la probabilidad de ese suceso:

$$S(E) = \frac{P(E)}{1-P(E)}$$

y

$$P(E) = \frac{S(E)}{1+S(E)}$$

12.2.4 Aproximaciones

(véanse también las páginas 227-234)

$$P(A \text{ y } B) = \text{mín}(P(A), P(B))$$

$P(A \text{ o } B) = \text{máx}(P(A), P(B))$.

Estas fórmulas no son, en rigor, correctas, y la magnitud del error que se comete con ellas depende del grado de independencia entre A y B; a falta de información sobre esta independencia, pueden resultar útiles.

$$\begin{aligned} \text{mín}(x,y) &= x, \text{ si } x \text{ es menor que } y \\ &= y, \text{ si } y \text{ es menor que } x \\ \text{máx}(x,y) &= x, \text{ si es mayor que } y \\ &= y, \text{ si } y \text{ es mayor que } x \end{aligned}$$

12.2.5 Combinaciones

(véanse también las páginas 45-46)

Dados n sucesos, de entre los que elegimos x, hay $\binom{n}{x}$ maneras de hacer la elección.

$$\binom{n}{x} = \frac{n!}{(n-x)!\ x!},$$

donde $n! = (n-1)(n-2)(n-3)\ldots(n-(n-1))$;
por ejemplo, $4! = 4.3.2.1 = 24$

Si hay n elementos de evidencia y cualquiera de ellos, o todos, pueden ocurrir, entonces hay

$$\sum_{x=0}^{x=n} \binom{n}{x}$$

combinaciones posibles de evidencia.

12.2.6 Estadísticas descriptivas.

(véanse también las páginas 143-148)

Las estadísticas descriptivas son, como su nombre indica, descripciones que normalmente se hacen para resumir las características principales de un conjunto de datos.

Media, m (también llamada promedio):

$$m = \sum_{i=1}^{i=n} \frac{x_i}{n}$$

Varianza, v:

$$v = \sum_{i=1}^{i=n} \frac{(x_i - m)^2}{n}$$

Desviación típica (o desviación estándar), s:

$s = SQR(v)$

12.2.7 Distribución normal

La mayoría de los métodos estadísticos que se usan en este libro son métodos no paramétricos. En la estadística no paramétrica no se presupone nada sobre el comportamiento subyacente de las variables que se estudian, y, en consecuencia, son menos propensos a fallar por causa de una suposición incorrecta.

La estadística paramétrica presupone modelos matemáticos subyacentes para las variables estudiadas. Una suposición frecuente es que las variables proceden de una población en la que siguen una función de distribución de probabilidad normal. O sea, la bien conocida curva en forma de campana. Como hay una teoría muy desarrollada para las distribuciones normales, resulta, con frecuencia, tentador el hacer uso de ella.

Pero, antes de hacerlo, es esencial examinar los datos para ver si en realidad están distribuidos normalmente. En la práctica, al contrario que en la teoría, muchas variables no se distribuyen normalmente, aunque a muchos estadísticos les gustaría que así fuera. Hay que resistir la tentación de aplicar la teoría basada en la distribución normal a variables que no están distribuidas normalmente.

12.2.8 Variables discretas y continuas

Una variable discreta es aquella que sólo puede tomar ciertos valores fijos; por ejemplo, las respuestas sí/no son discretas. Las variables con-

tinuas son las que pueden tener como valor cualquier número; por ejemplo, las cantidades de lluvia son continuas. Las variables continuas pueden hacerse discretas agrupando sus valores; por ejemplo, precipitación superior a 10 litros/m^2, precipitación inferior a 10 litros/m^2.

Si utiliza usted los mismos procedimientos de manipulación para las variables continuas y las discretas, seguramente se encontrará con problemas. Decida desde el principio el tipo de variables, y ajústese a él. En general, las variables discretas son mucho más fáciles de manejar que las continuas. Ello es debido, en parte, a que las variables continuas tienden a exigir métodos estadísticos paramétricos, y la elección de modelos paramétricos adecuados puede ser una cuestión problemática.

12.3 SUPERFICIES

(véanse también las páginas 60-65)

La ecuación general de una superficie es:

$$y = \sum_{i=0}^{i=n} b_i x_i$$

donde las b_i son constantes, y las x_i son variables.

El tablero de una mesa tiene una superficie bidmensional ($n = 2$), y x_1 y x_2 forman un ángulo recto. Esto puede expresarse diciendo que x_1 y x_2 son independientes entre sí, o que no están correlacionadas entre sí. En general, puede haber tantas variables como queramos, y pueden ser independientes unas de otras o no. Pueden utilizarse para describir algo que no está en el espacio real tridimensional. Pueden representar, por ejemplo, una determinada forma de comportamiento, o una enfermedad, o cualquier cosa.

Una superficie no es más que una entidad matemática.

12.4 DISCRIMINACIÓN

(véanse también las páginas 60, 76, 170)

Supongamos que tenemos n clases de objetos descritas por medidas realizadas sobre cierto número de variables. Entonces, se nos da un nuevo objeto y un conjunto de medidas de las variables para ese objeto,

y tenemos que decidir en cuál de las n clases hay que clasificar al objeto. Este es el problema de la discriminación.

En general, el problema sólo se puede resolver completamente si las distintas clases son linealmente separables. Es decir, si es posible colocar una superficie (o una serie de superficies) entre cada dos clases. Las superficies se definen en función de las variables que podemos medir en los objetos.

Podemos hacer que las clases sean mutuamente excluyentes, es decir, que el objeto sólo pueda clasificarse en una de las n clases; en este caso, lo colocaremos en la clase más probable, es decir, aquélla para la cual P(H:E) es máxima. Aquí, H define una clase, y E es toda la evidencia que relaciona a los objetos con su pertenencia a una u otra clase.

Obsérvese que, matemáticamente, P(H:E) es la ecuación de una superficie: es la superficie que «apunta» en la dirección de una determinada H. Los cálculos para obtener P(H:E) dan, indirectamente, los b_i de la superficie. Y la evidencia E sobre las variables da los x_i.

Sin embargo, no es obligatorio hacer uso explícitamente de las medidas de probabilidad. Cualquier método que produzca una superficie discriminante sirve. Cualquier método que funcione es correcto.

Si la superficie discriminante es:

$$y = \sum_{i=0}^{i=n} b_i x_i$$

podemos clasificar el objeto según que y supere o no algún valor y_c.

12.5 EL ALGORITMO DE APRENDIZAJE

(véanse también las páginas 51-53, 71, 78)

1. Tomar n observaciones, x_i, de un objeto a clasificar.

2. Calcular $$y = \sum_{i=0}^{i=n} b_{ik} x_i$$

   para cada una de todas las clases posibles.
   Inicialmente, $b_{ik} = 0$ para todos los i,k.

3. Encontrar la clase k para la cual y_k es el mayor.
4. Si el objeto pertenece a la clase k, entonces esta clasificación es correcta. No son necesarios cambios. Ir a 6.
5. Si la clasificación no es correcta, modificar b_{ik} del siguiente modo:
   $b_{ik} = b_{ik} + x_i$ para la clase k en la que el objeto tendría que haberse clasificado. Hacer esto para todos los b_{ik} de k.
   $b_{ik} = b_{ik} - x_i$ para todas las clases k en las que no habría que haber clasificado al objeto y en las cuales y_k es mayor que el y_k correspondiente a la clasificación correcta. Hacer esto para todos los b_{ik} de esas clases incorrectas.
6. Tomar otra observación. Ir a 1.

Este algoritmo precisa de un conjunto de objetos con clasificaciones conocidas para aprender y poder funcionar. Una vez que ya funciona bien, puede utilizarse para otros objetos cuyas clasificaciones no se conocen.

12.6 PROCEDIMIENTOS PARALELOS Y SECUENCIALES

(véanse también las páginas 83-89)

Un procedimiento paralelo toma toda la información disponible de una sola vez, y realiza un cálculo total sobre las base de esa información.

$$y = \sum_{i=0}^{i=n} b_{ik} x_i \qquad \text{para todas las k clases}$$

Un procedimiento secuencial va tomando una a una las variables, y va haciendo uso de esa información conforme avanza.

$$y_{ik} = y_{i-1,k} + b_{ik} x_i$$

Una vez que hubiera recopilado toda la información, un procedimiento secuencial daría el mismo resultado que un procedimiento paralelo. En el caso de que siempre se necesitase toda la información para obtener una conclusión, las diferencias entre ambos métodos serían puramente cosméticas.

Si puede llegarse a una conclusión sin utilizar toda la información posible, un procedimiento secuencial resulta ser más económico, porque llega antes a una conclusión.

12.7 VALORES MÍNIMOS Y MÁXIMOS

(véanse también las páginas 88, 92-93)

Si las variables x_i que se le dan al sistema llevan asociados valores máximos y mínimos, entonces se puede hacer que el procedimiento llegue más rápidamente a una conclusión.

Supongamos que todos los máx(x_i) fueran siempre a favor de una determinada clasificación, mientras que todos los mín(x_i) fueran en contra de ella. Entonces, si hay aún variables desconocidas, x_i, podemos calcular dos resultados posibles basados en que todas las variables pendientes tomen los valores máx(x_i) y mín(x_i). Si ninguno de esos resultados puede derrocar a la «conjetura óptima» actual del sistema, esa conjetura puede ya considerarse como la correcta, y, por consiguiente, no se necesitan realmente los valores de las x_i pendientes.

12.8 ESTRATEGIAS DE PROCESAMIENTO

(véanse también las páginas 174, 219-220)

Si se necesitan todas las x_i para hacer una clasificación, toda diferencia en las estrategias de procesamiento será simplemente cosmética. Si no se necesitan todas las x_i, entonces es importante utilizar la estrategia de procesamiento más eficaz posible con vistas a obtener una conclusión con la máxima eficacia.

12.8.1 Estrategias impulsadas por objetivos

El modo de funcionamiento impulsado por objetivos consiste en seleccionar una clase, y, con ella, comprobar las x_i adecuadas hasta que sea posible tomar una decisión sobre esa clase.

Hecho eso, el sistema puede pasar a comprobar la siguiente clase, y así sucesivamente.

12.8.2 Estrategias impulsadas por datos

Consisten en seleccionar alguna x_i que, en algún sentido, parezca que es útil conocer. Obtenido un valor para esa x_i, el sistema hace todo el uso que puede de esa información.

Hecho eso, el sistema selecciona otra x_i. A lo largo de este proceso, se van alcanzando conclusiones sobre distintas clasificaciones.

12.8.3 Selección de la siguiente variable
(véanse también las páginas 93, 245)

Con indiferencia de que el sistema funcione con una estrategia impulsada por objetivos o con una impulsada por datos, normalmente existe cierta libertad en relación con la elección de la siguiente variable a considerar. El problema consiste en escoger, de entre todas las preguntas posibles que pueden plantearse, una que sea «buena», y es difícil precisar aquí qué es «buena». Sin embargo, para cualquier pregunta, podemos considerar:

1. ¿Sobre cuántas clasificaciones influye?
2. ¿En qué medida influye en esas clasificaciones?

Por ejemplo:

1. Para cada una de las x_i actualmente desconocidas, podríamos calcular:

   y_{ik} (utilizando máx x_i)—y_{ik} (utilizando mín x_i)
   para estimar el máximo cambio posible que puede proporcionar el conocimiento de x_i.

2. Utilizando probabilidades bayesianas, podríamos calcular:
   P(H:E)—P(H:no E),
   donde E es la observación x_i, para estimar el máximo cambio que puede proporcionar cada elemento de evidencia.

El calcular esos elementos para cada variable antes de hacer una pregunta, implica más procesamiento (por parte del ordenador) que el plantear simplemente la siguiente pregunta de una lista. La ventaja es que el trabajo total que tiene que hacer la persona que usa ese ordenador puede que sea menor.

He aquí, sin embargo, algunos trucos para reducir el tiempo de procesamiento:

Pruebe con máx(máx(x_i)—mín(x_i)) para tener una idea de la «importancia» de cada pregunta. La pega con este método es que puede suceder que x_i tenga un margen de variación muy grande, y que, sin embargo, no sea nada importante.

Pruebe con máx(varianza(b_{ik})) para tener una idea de la medida en que cada variable se utiliza en las clasificaciones. La pega es que puede ocurrir que una variable que parezca utilizarse mucho tenga en realidad un valor pequeño de (máx(x_i)—mín(x_i)), y que justamente por ello sea grande la variabilidad de b_{ik}.

12.9 CONCLUSIONES INTERMEDIAS

(véanse también las páginas 155-157)

Las conclusiones intermedias no siempre son estrictamente necesarias, pero pueden resultar útiles para «humanizar» al sistema. Una conclusión intermedia puede utilizarse como variable de entrada a otra etapa en el proceso del experto. El uso de conclusiones intermedias hace mucho más difícil la escritura inicial del sistema.

Sin embargo, hay algo que merece la pena recordar con respecto a las conclusiones intermedias, algo que gira en torno a la fastidiosa cuestión de la independencia.

La mayoría de los métodos estadísticos utilizados supone que los distintos elementos de evidencia E_i son independientes, y, con frecuencia, estas hipótesis de independencia no están justificadas.

Si hay n elementos de evidencia que apoyan a cierta hipótesis H y están correlacionados entre sí, pero se hacen los cálculos como si tales correlaciones no existieran, entonces va a ser como si H recibiera más apoyo del que realmente tiene.

La inserción de conclusiones intermedias en el proceso de razonamiento, puede ayudar a eliminar este efecto.

Supongamos que están presentes E_1 y E_2, ambos apoyando a la hipótesis H_1, y que ésta es una conclusión intermedia que sirve de evidencia para otra conclusión posterior, H_2.

Supongamos también que, en alguna medida, E_1 y E_2 están correlacionados entre sí. Entonces, H_1 recibe más apoyo del que debería.

Podría ocurrir, sin embargo, que, aún así, fuera razonable decir que H_1 es cierta, y en ese caso no estaríamos introduciendo ninguna falsedad en esta etapa.

Pero, si no existiera H_1, el error debido a la correlación entre E_1 y E_2 se propagaría hacia H_2, y se iría haciendo cada vez más grave conforme

se fueran introduciendo en los cálculos otros E_i con algún grado de correlación con todos los anteriores.

La presencia de H_1 y de otras conclusiones intermedias permite hacer, por así decirlo, borrón y cuenta nueva. Puede hacerse un nuevo conjunto de cálculos para H_2, basado en un número menor de elementos de evidencia, reduciéndose así el riesgo de una acumulación peligrosa de incertidumbre.

12.9.1 Sistemas explicativos

(véanse también las páginas 210-213)

Las conclusiones intermedias pueden utilizarse para dar explicaciones sobre el estado del sistema en cada momento.

En general, un sistema explicativo puede decir hasta dónde ha llegado en su proceso de razonamiento, y también, con frecuencia, cómo ha llegado hasta allí. El problema es que ello obliga a que el ordenador funcione de tal modo que pueda tener las explicaciones preparadas, lo cual es difícil de conseguir, y puede conducir a una realización menos eficaz.

Por ejemplo, ¿qué tipo de explicación sería la adecuada si el ordenador preguntara por x_i, debido a que previamente ha calculado que esta x_i en particular tiene más influencia sobre P(H:E) que cualquiera otra x_i? La mayoría de las estrategias presentadas en este libro para seleccionar una «buena» x_i se basan en buscar una x_i que parezca importante desde un punto de vista matemático. Pero un usuario normal quizá se sintiera más cómodo con explicaciones que dieran, por ejemplo, un listado de las clasificaciones más afectadas por esa x_i, independientemente de por qué realmente el sistema escogió tal x_i.

12.10 INTERPOLACIÓN LINEAL DE RESPUESTAS

(véanse también las páginas 234, 249)

Supongamos que el usuario no está seguro sobre la respuesta que debe dar a una pregunta del sistema. Puede venirle bien dar esa respuesta como un valor dentro de una cierta escala, que vaya, por ejemplo, de menos 5 a más 5, es decir, entre «No» (—5), «No sé» (0) y «Sí» (+5).

Supongamos también que la probabilidad *a priori* de una respuesta cualquiera es P(E).

Si hay incertidumbre en la respuesta, el sistema tendrá que hacer

frente a ella poniendo cierto valor en P(E) y otro en P(no E). Si el usuario no está seguro sobre la existencia de un cierto elemento de evidencia, podría ser que tal elemento no estuviera presente.

Sea R la respuesta del usuario. Si R es mayor o igual que cero, entonces

$$P(E) = P(E) + (1 - P(E))R/5$$

y si es menor o igual a cero,

$$P(E) = P(E) + P(E)R/5$$

Evidentemente, P(no E) = 1—P(E). Si R = 0, P(E) permanece con su valor anterior.

Entonces, para calcular la nueva P(H:E), haremos:

$$P(H:E) = P(H:E)P(E) + P(H:\text{no } E)P(\text{no } E)$$

Dicho de otro modo, se calculan los dos resultados, y se obtiene el resultado final ponderando cada uno de ellos, de acuerdo con la certidumbre que ha expresado el usuario a favor y en contra de la evidencia.

12.11 FORMATOS DE LOS DATOS

1. Matrices

DIM R(E,H)

	Hipótesis 1	Hipótesis 2	... etc.
Evidencia 1	x	x	
Evidencia 2	x	x	
... etc.			

2. Sentencias DATA (véase también el apartado 10.1).

DATA hipótesis 1, probabilidad *a priori* de la hipótesis 1, evidencia j, $P(E_j:H)$, $P(E_j:\text{no } H)$

DATA hipótesis 2, etc.

y:

DATA j, evidencia j

La matriz R mantiene un conjunto de ecuaciones de superficies que permiten llevar a cabo el proceso de clasificación.

Las sentencias DATA llevan una información similar: cada una de ellas es equivalente a una columna de la matriz.

La ventaja del uso de matrices es que el procesamiento es más rápido, pero los datos pueden perderse si no se toma la precaución de guardarlos.

La ventaja del uso de sentencias DATA es que no se pierden tan fácilmente, pero el proceso puede resultar más lento.

Un buen método puede ser el de guardar la información en un fichero del disco, si es que dispone usted de unidad de disco.

13. SELECCIÓN BIBLIOGRÁFICA

Resulta bastante problemático dar una lista de libros recomendados para un tema tan difuso como el de los sistemas expertos, porque lo mejor sería leer sobre el asunto en el que uno quiere que su sistema sea experto, y esto cubriría potencialmente todo.

Por otra parte, si la lista es demasiado larga, nadie se animará a leer todo lo que contiene, y si es demasiado corta omitirá muchas cosas. A pesar de todo (y en orden alfabético)...

Bailey, N.J.T. Mathematics, Statistics and Systems for Health. Wiley, 1977.

Incluye una parte interesante sobre diagnóstico médico. Gracias a su orientación médica, no se restringe a métodos teóricamente «perfectos», sino que se concentra en métodos que funcionan.

Higman,B. A Comparative Study of Programming Languages. McDonald/Elsevier Computer Monographs, 1967.

Un libro relativamente barato, que puede servir para abrir los ojos a quien piense que sólo hay tres o cuatro lenguajes de programación diferentes. Como la mayoría de los libros que cubren un gran número de lenguajes, puede causar indigestión mental si se trata de leer demasiado rápidamente.

Hunt,E.B. Artificial Intelligence. Academic Press, 1975.

Un buen libro sobre inteligencia artificial, aunque puede resultar algo abstracto para un lector con poca base matemática. Incluye el algo-

ritmo de aprendizaje y la inferencia bayesiana, y una amplia gama de temas interesantes.

Kendall, M. Multivariate Analysis. Charles Griffin, 1975.

Este es, ciertamente, uno de los favoritos de siempre de este autor. Es un libro sobre estadística, que cubre los problemas de clasificación y distancia. De ningún modo es el libro más actualizado (de hecho, cuando habla de ordenadores, sugiere varias veces que algún día se generalizará el uso de las pantallas), pero contiene una gran dosis de sentido común, algo difícil de encontrar en este tema.

Knuth, D.E. The Art of Computer Programming, Volume 1: Fundamental Algorithms. Addison-Wesley, 1973.

Es prácticamente imposible no tener un ejemplar de este libro. Es indispensable si quiere uno divertirse con encadenamientos y estructuras de árboles, aunque su gran inconveniente es que todos los ejemplos se dan en MIX, una especie de lenguaje ensamblador para una máquina ficticia.

Michie, D. (Ed.). Expert Systems in the Microelectronic Age. Edinburgh University Press, 1979

Una útil colección de artículos actuales sobre varios sistemas, como PROSPECTOR, MYCIN, PUFF, SU/X, MECHO y otros. Su utilidad principal está en que revela cuán complicadas pueden llegar a ser las cosas.

Mood, A.M., Graybill, F.A., Boes, D.C. Introduction to the Theory of Statistics. McGraw-Hill, 1974.

Son tantos los aspectos de los sistemas expertos relacionados con la estadística, que el tener un libro sobre ese tema es esencial. La pega es que tales libros no pueden, naturalmente, restringirse solamente a lo que a uno le interesa saber, sino que le dan todo un curso sobre todos los aspectos de la estadística. En cualquier caso, este libro cubre la inferencia bayesiana y los conceptos básicos de probabilidades.

Morrison, D.F. Multivariate Statiscal Methods. McGraw-Hill, 1976.

Es un libro alternativo al de Kendall. Algo árido, pero usa notación

matricial, en contraste con el uso que hace Kendall de variables con subíndices. Según el lector, uno u otro le parecerá más fácil. Con los lenguajes de programación, ambos utilizan matrices.

Robinson, J.A. Logic: Form and Function. Edinburgh University Press, 1979.

A algunos lectores puede resultarles éste un libro difícil, si no les gusta demasiado la lógica, pero tiene una parte muy buena sobre LISP, y, en cualquier caso, si quiere usted ir hacia sistemas más complejos, le va a ser muy difícil evitar la lógica formal.

Wani, J.K. Probability and Statistical Inference. Appleton-Century-Crofts, 1971.

Otro libro más de estadística, que también incluye a Bayes y a las probabilidades.

Es una buena idea, siempre que sea posible, tener al menos dos libros sobre cualquier tema, especialmente si el tema es difícil.

De ese modo, puede usted leer una versión, no comprenderla, leer otra versión, tampoco comprenderla, y luego volver a leer la primera versión, momento en el cual la luz puede comenzar a hacerse.

Según eso, y puesto que ya dispone usted de un libro (éste), ¿cuál es el otro libro hacia el que debe usted precipitarse y comprarlo?.

Probablemente, el de Kendall. No es en sí un libro sobre ordenadores, pero es una maravillosa fuente de ideas para alguien que tenga un ordenador. O, si no lo encuentra, el de Hunt.

(N. del T.) En el tiempo transcurrido entre la edición original de este libro y la presente traducción, se han publicado varias obras sobre el tema específico de los sistemas expertos. Referenciamos a continuación las que conocemos:

Alty, J. y Commbs, M. Expert Systems-Concepts and Examples. NCC Books y Wiley, 1984.

Buchanan, B.G. y Shortliffe, E.H. Rule-based Expert Systems. Addison-Wesley, 1984.

Clancey, W. y Shortliffe, E.H. Readings in Medical Artificial Intelligence: The First Decade, Addison-Wesley, 1984.

Coombs, M.J. Developments in Expert Systems, Academic Press, 1984.

Fieschi, M. Intelligence Artificielle en Médecine: des Systèmes Experts. Masson, 1984.

Forsyth, Expert systems. Principles and Case Studies. Methuen, 1984.

Gondran, M. Introduction aux Systèmes Experts. Eyrolles, 1984.

Hayes-Roth, F. Waterman, D.A. y Lenat, D.B. Building Expert Systems. Addison-Wesley, 1983.

Michie, D. Introductory Readings in Expert Systems. Gordon and Breach, 1984.

Torsun, I.S. Expert Systems. State of the Art. Addison-Wesley, 1983.

Weiss, S.M. y Kulikowski, C.A. A practical Guide to Designing Expert Systems. Rowman & Allanheld, 1984.

INDICE DE PALABRAS

AGE, 236
Algoritmo de aprendizaje, 54, 65, 71, 78, 288
AL/X, 236
Antecedente-consecuente, 217
Aplicaciones, 199

Base de conocimientos, 27-33
BASIC, 216, 274-280

CASNET, 236
Clasificación, 60, 65, 142, 287
Combinaciones, 46, 285
Conectivas lógicas, 171

DENDRAL, 222-226
Desviación típica, 144, 286
Discriminación, 65, 287
Distribución normal, 144, 286
Dominio de consulta, 28

EL, 236
EMYCIN, 214
Encadenamiento hacia adelante, 165, 219
Encadenamiento hacia atrás, 207, 220
Espacio de descripción, 60, 65
EXPERT, 236

Factor de certidumbre, 206
Factorial, 46, 124
Función de distribución de probabilidad (fdp)., 144

GUIDON, 213

HEARSAY, 236

IF...THEN..., 25, 206, 216
INTERNIST, 236

Ji-cuadrado, 118, 120

KAS, 234, 236

Lenguaje de máquina, 274
LISP, 274-280

MECHO, 236, 238
Media, 67, 143, 286
Medidas de distancia, 68, 141
META-DENDRAL, 225
Micro-Expert, 236
MOLGEN, 236
MYCIN, 205-214

PECOS, 236
PIP, 236
Posibilidades, 284
Probabilidad lógica, 206
Probabilidades, 33, 36, 69, 145, 205, 229, 281
Probabilidades *a posteriori*, 283
Probabilidades *a priori*, 69, 283
Probabilidades condicionales, 36, 37, 282
Probabilidades conjuntas, 36, 39, 282
Procedimientos paralelos, 83, 289
Procedimientos secuenciales, 84, 289
PROLOG, 275
PROSPECTOR, 227-235
PUFF, 214-222

Recursión, 220, 279
Reglas de producción, 25, 205, 216
ROSIE, 236
R1, 236

SACON, 236
SAGE, 236
SECS, 236
Separabilidad lineal, 75
Similaridad, 146
Sistemas adaptativos (véase también Algoritmo de aprendizaje), 24
SOPHIE, 236

Sucesos independientes, 45, 282
SU/X, 236
SINCHEM, 236

TEIRESIAS, 211
Teorema de Bayes, 47, 230, 282

Valores máximos y mínimos, 88, 97, 290
Variables continuas y discretas, 286
Varianza, 96, 145, 286
Varianza unidad, 145
VM, 236

Apéndice a la edición española

La inclusión de este apéndice ya se ha justificado en la «Nota preliminar del traductor». Los algoritmos admiten muchas mejoras, pero nos ha parecido mejor conservar su esencia y dejar los perfeccionamientos ál buen juicio del lector. Para ello, le resultará mucho más fácil partir de estos programas en Pascal (aun en el caso, como decíamos en la nota, de que quiera o tenga que seguir programando en BASIC), en los que las estructuras de los algoritmos quedan reflejadas de una manera mucho más nítida. El que la longitud de estas versiones en Pascal sea muy superior a la de los programas originales en BASIC le resultará natural al lector que conozca las diferencias entre ambos lenguajes (aunque también contribuye algo a ello el hecho de haber añadido ciertas facilidades conversacionales). Algunos detalles seguramente podrían haberse programado en Pascal de una manera más racional, pero hemos optado por no alejarnos en exceso de los programas originales.

No hemos utilizado el «Pascal estándar», sino una versión llamada «Turbo Pascal»*, disponible para microordenadores con sistemas operativos CP/M y MS/DOS. Esta versión incluye algunas características, como la posibilidad de definir «strings» y operar sobre ellos, y diversas facilidades de manejo de ficheros y de periféricos, características que también se encuentran en otras versiones de Pascal para micros, como «Pascal/M» y «Pascal/MT + »*.

---

* «Turbo Pascal», «Pascal/M» y «Pascal/MT + » son nombres comerciales registrados, respectivamente, por Borland International, Sorcim y Digital Research.

El experto multinodo

El listado que se reproduce a continuación es una versión del sistema descrito al final del Capítulo 7. En previsión de posibles limitaciones de memoria, hemos optado por mantener en la memoria central en cada momento las informaciones relativas a un solo nodo, por lo que es preciso acceder a disco para recorrer todos los nodos. Ello hace más lenta la ejecución, pero el programa cabe perfectamente en sistemas con sólo 32 KB de RAM.

```
PROGRAM experto_aprendiz(input,output,BC);


CONST
   MaxEnt =20;  MaxSal=20;   MaxNodo =50;
   MaxEjem=50;  MaxCad=30;   MaxOpcion=8;
TYPE
   Ient      = 1..MaxEnt;
   Isal      = 1..MaxSal;
   Inodo     = 1..MaxNodo;
   Iejem     = 1..MaxEjem;
   Iopcion   = 1..MaxOpcion;
   Cadena    = String[MaxCad];
   NomFich   = String[14];
   Decision  = Array[Isal] of real;
   Variable  = Record
                 Nombre        : cadena;
                 Valor,
                    Vmax,Vmin  : real;
                 Marca         : boolean
                 end;
   Resultado= Record
                 Nombre        : cadena;
                 Marca         : boolean
                 end;
   Ejemplo   = Record
                 ValEnt        : array[Ient]
                                    of real;
                 ValSal        : Isal
                 end;
```

```
   Nodo      = Record
                 NumEnt        : Ient;
                 NumSal        : Isal;
                 NumEjem       : Iejem;
                 Entrada       : array[Ient]
                                 of Variable;
                 Salida        : array[Isal]
                                 of Resultado;
                 NodEjem       :array[Iejem]
                                 of Ejemplo;
                 NodRegla :array[Ient,Isal]
                                 of real
                 end;
   BaseConoc= file of Nodo;

VAR
     F                    : NomFich;
     N, Contnodo          : Inodo;
     NumResult            : Isal;
     Opcion               : Iopcion;
     Resp, NoDec,
        MultiNodo         : boolean;
     VectorDecision       : Decision;
     VarNodo,
        VarNodoP          : Nodo;
     BC                   : BaseConoc;

PROCEDURE buscafich
 (VarNomFich:NomFich; var existe:boolean);

{Devuelve existe = true o false segun
que el nombre que se le da corresponda o
no a un fichero existente en el disco}

BEGIN
assign(BC,VarNomFich);
{$I-} reset(BC); {$I+}
existe:=(IOresult=0)
END;

PROCEDURE leeentero
 (var numero:integer; maximo:integer);
```

```
var resultado:boolean;
BEGIN
  repeat
  {$I-}  readln(numero); {$I+}
  resultado:=(IOresult=0);
  if not resultado then
    writeln('Eso no es un numero entero.',
            ' Escribelo otra vez');
  if numero > maximo then
    write('Valor maximo = ',maximo:3,
          'dame  otro valor');
  if numero < 1 then
    begin
      write('Espero un numero entero');
      writeln(' y mayor que cero');
      write('Dejate de tonterias ');
      writeln('y escribelo bien')
    end
  until resultado and (numero<=maximo)
                  and (numero>0);
 writeln
END;
PROCEDURE leereal(Var numero: real);

var resultado:boolean;
BEGIN
  repeat
  {$I-} readln(numero); {$I+}
  resultado:=(IOresult=0);
  if not resultado then write
    ('Error de formato; repite el dato  ')
  until resultado
END;

PROCEDURE pregunta(var SiNo:boolean);

var car:char;

BEGIN
repeat
 readln(car);
```

```
 if car in ['s','S']
    then SiNo:=true
    else
     if car in ['n','N']
      then SiNo:=false
      else
       write('Tienes que pulsar S o N   ')
until car in ['s','n','S','N']
END;

PROCEDURE menu(var op:Iopcion);

{Muestra el menu y lee la opcion elegida}

var Num: integer;

BEGIN
limpiar;
writeln('1. Inicializar un experto');
writeln('2. Introducir ejemplos');
writeln('3. Ejercitar al experto');
writeln('4. Sesion de entrenamiento');
writeln('5. Funcionamiento normal');
write('6. Designar un fichero donde');
writeln(' se encuentra un experto');
write('7. Examinar base de ');
writeln('conocimientos y ejemplos');
write('8. Cerrar fichero, y, si se');
writeln(' quiere, terminar');
writeln;
write('Elije una opcion por su numero  ');
leeentero(Num,MaxOpcion);
op:=Num
END;

PROCEDURE limpiar;

BEGIN
ClrScr;
write('SOY EL SISTEMA EXPERTO MULTINODO');
writeln(' CON APRENDIZAJE');
write('_____________________________________');
```

```
writeln('_________________');
writeln
END;

PROCEDURE inicia;

{Abre el fichero y pide el numero de nodos
 y los nombres de las variables y de los
 resultados}

var i                : Ient;
    j                : Isal;
    no               : Inodo;
    NomVar           : cadena;
    respuesta,
    sigue,
    indicador        : boolean;
    maximo,minimo    : real;
    Num              : integer;
    nfich            : NomFich;

    procedure comprueba
      (var indicador,sigue:boolean;
       p:Inodo);

    {Mira en los nodos anteriores si el
     nombre de la variable
     se ha definido previamente}

    var pp:Inodo; qq:Ient;

    begin
    reset(BC);
    repeat
    respuesta:=false;
    pp:=0;
    write('Espera un momento: estoy')
    writeln(' mirando las entradas ');
    writeln('del nodo',pp+1:3);
    read(BC,VarNodoP);

    for qq:=1 to VarNodoP.NumEnt do
```

```
        with VarNodoP.Entrada[qq] do
        if (Nombre=NomVar)
         then
           begin
           write('Nombre ya dado a una');
           writeln(' variable en el nodo'
                                  ,pp+1:3);
           write('con valor maximo = '
                              ,vmax:10:3);
           write(' y valor minimo = '
                              ,vmin:10:3);
           write('.  Es la misma?    ');
           pregunta(respuesta);
           if respuesta
            then
              begin
              indicador:=false;
              minimo:=vmin;
              maximo:=vmax
              end
             else
              begin
              write('Entonces dame ');
              writeln('otro nombre');
              sigue:=false
              end
           end;  {if Nombre=NomVar}
    pp:=pp+1
    until (pp=p)
          or respuesta
    end;

BEGIN  {inicia}

limpiar;
repeat
   write('Nombre del fichero en el que ');
   writeln('va a residir el experto?');
   readln(nfich);
   buscafich(nfich,indicador);
   if indicador
```

```
     then
       begin
       write('Ese fichero ya existe. ');
       write('Quieres borrarlo?   ');
       pregunta(respuesta)
       end
until respuesta or not indicador;
rewrite(BC);
write('Cuantos nodos va a tener ');
write('el experto  ?');
leeentero(Num,MaxNodo);
N:=Num;
MultiNodo := N>1;
for no:=1 to N do
  begin
   limpiar;
   VarNodo.NumEjem:=0;
   write('Cuantas variables de entrada');
   write(' en el nodo',no:3,'?   ');
   leeentero(Num,MaxEnt);
   VarNodo.NumEnt:=Num;
   writeln('Bautizalas:');
   for i:=1 to Num do

    begin

     repeat
      write('Nombre de la variable',i:3);
      write(' en el nodo',no:3,'?   ');
      readln(NomVar);
      sigue:=true;
      indicador:=true;
      if no>1 then
          comprueba(indicador,sigue,no-1)
     until sigue;

     if indicador then
        begin
          write('Cual es su valor');
          write(' maximo?   ');
          leereal(maximo);
```

```
      write('Y su valor minimo?   ');
      leereal(minimo)
    end;

  with VarNodo.Entrada[i] do
    begin
      Valor:=0;
      Nombre:=NomVar;
      Vmax:=maximo;
      Vmin:=minimo;
      Marca:=true
    end

end  {for i};
with VarNodo do

   begin
     write('Cuantos resultados en');
     write(' el nodo ',no:3,'?   ');
     leeentero(Num,MaxSal);
     NumSal:=Num;
     writeln('Bautizalos:');
     for j:=1 to NumSal do
     with Salida[j] do
      begin
       write('Nombre del resultado ');
       write(j:3,'?   ');
       readln(Nombre);
       marca:=true;
       for i:= 1 to NumEnt do
                  NodRegla[i,j] := 0
      end
   end;   {with VarNodo}

if no>1 then

   begin
     seek(BC,FileSize(BC));
     write(BC,VarNodo)
   end
```

```
  end    {for no}
END;
PROCEDURE inicmarcas;

{Pone las marcas que indican que aun no
 se conocen los valores de las variables
 y que todos los resultados son posibles}

var No: Inodo;

    procedure anula;
    var  i:  Ient;
         j:  Isal;
    begin
    with VarNodo do
         begin
         for i:=1 to NumEnt do
         with Entrada[i] do
          begin
          Valor:=0;
          Marca:=true
          end;
         for j:=1 to NumSal do
          with Salida[j] do Marca:=true
         end
    end;

BEGIN
if MultiNodo then
   begin
   reset(BC);
   for No:=1 to N do
       begin
       read(BC,VarNodo);
       anula;
       seek(BC,No-1);
       write(BC,VarNodo)
       end   {for No}
   end
else anula
END;
```

```
PROCEDURE pidevalor
 (numero:Ient; var ValVar:real);

{Pide el valor de la variable cuyo numero
 se le indica en "numero", lo lee como
 numero real, comprueba si el valor leido
 esta dentro del rango dado en la inicia-
 lizacion hace "false" la variable global
 "Marca" correspondiente a esa variable
 para no preguntar mas por ella, y entrega
 el valor leido en ValVar}

var vale: boolean;

BEGIN

vale:=false;
with VarNodo.Entrada[numero] do
   begin
     write('Que valor tiene la ');
     write('variable "',Nombre,'"?  ');
     repeat
        leereal(ValVar);
        if (ValVar>Vmax) or (ValVar<Vmin)
         then
          begin
          write('Valor fuera del rango ');
          writeln('dado, que es:');
          write('Vmin =',Vmin:10:3);
          write(';  Vmax =',Vmax:10:3);
          write('.   Dame otro')
          end
         else vale:=true
     until vale;
     Marca:=false
   end

END;
PROCEDURE deducir
 (No:Inodo; var D: Decision; var hj:Isal;
  var NoDecision: boolean);
```

```
{Va decidiendo cual es la mas importante
 de las variables que aun no se conocen
 y pidiendo sus valores hasta poder
 calcular el numero, hj, del resultado
 mas plausible en el nodo No. Si no
 puede llegar a tomar una decision, hace
 NoDecision = true }

var  hi,j,k   : Isal;
     ii,vv    : Ient;
     PD       : array[Isal] of real;
     busca,
         vale: boolean;

   procedure calculdec(vv:Ient);

   {Actualiza los valores del vector de
    decision, D, teniendo el cuenta el
    valor de la nueva variable de entrada,
    vv, que sa ha introducido con el
    procedimiento pidevalor}

   var jj:Isal;
   begin
      with VarNodo do
        for jj:=1 to NumSal do
         begin
          D[jj]:=D[jj]+
            Entrada[vv].Valor
              *NodRegla[vv,jj];
          PD[jj]:=D[jj]
         end
   end;    {calculdec}
   procedure puntuavar(var vv:Ient);

   {Busca cual es la variable que mas
    puede influir en la decision,
    y devuelve su numero por vv}

   var VR : array [Ient] of real;
       i  : Ient;
```

```
    j   : Isal;
    n1  : Integer;
    m,h: real;
begin
   h:=0; vv:=0;
   with VarNodo do
     for i:=1 to NumEnt do
       begin
       VR[i]:=0; m:=0; n1:=0;
       for j:=1 to NumSal do
        begin
         if Entrada[i].Marca
           and Salida[j].Marca
           then m:= m + NodRegla[i,j];
         if Salida[j].Marca then
                                  n1:=n1+1
         end;
       if n1>0 then m:= m/n1;
       for j:= 1 to NumSal do
          if Entrada[i].Marca
            and Salida[j].Marca
            then VR[i]:= VR[i] +
              abs(Entrada[i].Vmax-
                  Entrada[i].Vmin)*
                  sqr(NodRegla[i,j]-m);
       if VR[i] > h then
          begin
          vv:=i;  h:=VR[i]
          end
       end {with VarNodo, for i}
end;        {puntuavar}
procedure decidir
           (vv: Ient; var hj: Isal;
                   var busca: boolean);

{Decide si la informacion disponible ya
 es suficiente para tomar una decision,
 sin necesidad de pedir mas valores}

 const MaxReal= 1E+37;
 var   i       :Ient;
```

```
    hi,j    :Isal;
    h2,dif  :real;

procedure calculresult
                        (var nmax:Isal);
{Obtiene el numero del resultado al
 que corresponde el maximo valor en
 el vector de decision, D[i]}

var jj,nn:Isal;
    dmax  :real;

begin

  dmax:=-MaxReal;
  nmax:=0;
  nn:=VarNodo.NumSal;
  for jj:=1 to nn do
      if D[jj]>dmax then
                  begin
                   nmax:=jj;
                   dmax:=D[jj]
                  end

end;    {calculresult}
begin      {decidir}
 calculdec(vv);
 calculresult(hj);
 if hj>0 then with VarNodo do
  begin
   for i:=1 to NumEnt do
    for j:=1 to NumSal do
     if Entrada[i].Marca then
      begin
       dif:=NodRegla[i,j]
             -NodRegla[i,hj];
       if dif>0 then PD[j]:=PD[j]+
                     dif*Entrada[i].Vmax
                 else PD[j]:=PD[j]+
                     dif*Entrada[i].Vmin
      end;
```

```
        h2:= -MaxReal;
        for j:=2 to NumSal do
         begin
          if PD[j] >= h2 then
           begin
            h2 := PD[j];
            hi := j
           end;
          if Salida[j].Marca and
             (PD[j] < D[hj]) and (j <> hj)
            then
             begin
              Salida[j].Marca:=false;
              write('No puede ser el ');
              write('resultado ');
              writeln
                (VarNodo.Salida[j].Nombre)
             end
         end;
        if PD[hi] <= PD[hj] then
                                busca := false
      end   {if hj>0, with VarNodo}
   end;   {decidir}
BEGIN     {deducir}

   if Multinodo then
     begin
       limpiar;
       writeln('Nodo', No:3)
     end;
   busca:=true;
   NoDecision:=true;
   with VarNodo do

     begin
      for ii:=1 to NumEnt do
        begin
         D[ii]:=0;
         PD[ii]:=0;
         if not Entrada[ii].Marca then
                                  calculdec(ii)
```

```
        end;
      while busca do
        begin
         puntuavar(vv);
         if vv > 0 then

          begin
           Entrada[vv].Marca := false;
           vale := false;
           for j:=1 to NumSal-1 do
             for k:=j+1 to NumSal do
              if (NodRegla[vv,j] <>
                     NodRegla[vv,k])
                 and Salida[j].Marca
                 and Salida[k].Marca
                    then vale := true;
            if vale then
              begin
                pidevalor
                  (vv,Entrada[vv].Valor);
                decidir
                  (vv,hj,busca)
              end
           end  {caso vv>0}

          else {caso vv=0}
               busca:=false
        end   {while busca}
      end {with VarNodo};
    if hj > 0 then
      begin
          NoDecision:=false;
          writeln('Deduzco el resultado: ',
                 VarNodo.Salida[hj].Nombre)
      end
            else
             begin
               write('No tengo elementos');
               writeln(' de juicio')
          end
END {deducir};
```

```
PROCEDURE actualizar
  (No:Inodo; result:Isal);

{Tras haber trabajado con el nodo No,
 cuya salida se le da en "result",
 se llama a este procedimiento, que
 recorre todos los otros nodos para ver
 si hay informacion que sirva para ellos:
 entradas coincidentes con alguna de las
 entradas o de las salidas de No}

var p        : Inodo;
    v,vp     : Ient;
    r,q      :  Isal;
    modifica: boolean;
BEGIN
write('Un momento: estoy mirando los ');
write('otros nodos para ver si ');
writeln('tengo informacion para ellos');
seek(BC,No);
for p:= No+1 to N do
 begin
  read(BC,VarNodoP);
   modifica:=false;
   for vp:=1 to VarNodoP.NumEnt do
    begin
     for v:=1 to VarNodo.NumEnt do
      if (not VarNodo.Entrada[v].Marca)
         and
         (VarNodo.Entrada[v].Nombre=
          VarNodoP.Entrada[vp].Nombre)
        then
         begin
          VarNodoP.Entrada[vp].Valor:=
              VarNodo.Entrada[v].Valor;
          VarNodoP.Entrada[vp].Marca:=
              false;
          modifica:=true
         end;
    if VarNodoP.Entrada[vp].Nombre =
         VarNodo.Salida[result].Nombre
```

```
          then
           with VarNodoP.Entrada[vp] do
            begin
             Valor:=vmax;
             Marca:=false;
             modifica:=true
            end;
       for r:=1 to VarNodo.Numsal do
         if (VarNodoP.Entrada[vp].Nombre=
               VarNodo.Salida[r].Nombre)
              and (r<>result)
           then
            begin
             VarNodoP.Entrada[vp].Marca:=
                                        false;
             modifica:=true
            end

    end;  {for vp}
   if modifica then
            begin
              seek(BC,p-1);
              write(BC,VarNodoP)
            end

 end;  {for p}

seek(BC,No)

END;

PROCEDURE ejemplos;

{Introduce ejemplos en los N nodos}

var N2,i: Iejem;
    No:   Inodo;
    j:    Ient;
    k:    Isal;
    Num:  integer;
    respuesta: boolean;
```

```
BEGIN

 if MultiNodo then
                reset(BC);
 for No:=1 to N do
  begin
   limpiar;
   writeln('Nodo', No:3);
   if MultiNodo then
                 read(BC,VarNodo);
   write('Cuantos ejemplos?   ');
   leeentero(Num,MaxEjem);
   N2:=Num;
   if N2>0 then
    begin
     with VarNodo  do
      begin
       if NumEjem>0 then
        begin
         write('Quieres anular los ');
         write('anteriores ejemplos?   ');
         pregunta(respuesta);
         if respuesta then
                       NumEjem:=0
        end;
        for i:=NumEjem+1 to NumEjem+N2 do

         begin
          limpiar;
           write('Ejemplo',i:3,' en el ');
           writeln('nodo',No:3);
           for j:=1 to NumEnt do
             pidevalor
                  (j,NodEjem[i].ValEnt[j]);
           writeln('Resultados posibles:');
           for k:=1 to NumSal do
             writeln
                (k:3,'.',Salida[k].Nombre);
           write('Dime el numero ');
           write('del correcto   ');
           leeentero(Num,MaxSal);
```

```
          NodEjem[i].ValSal:=Num
        end  {for i} ;

       NumEjem:=NumEjem+N2

      end;    {whith}

     if MultiNodo then
         begin
           seek(BC,No-1);
           write(BC,VarNodo)
         end

    end      {if N2>0}

  end {for No}

END;
PROCEDURE ejercitar;

{Construye reglas a base de trabajar
repetidamente con los ejemplos que tiene}

var i,p  : integer;
    d,d2: real;
    v,j  : Ient;
    q,k  : Isal;
    No   : Inodo;
    c    : integer;

BEGIN

limpiar;
write('Espera un momento: ');
writeln('estoy haciendo ejercicios');
randomize;
if MultiNodo then reset(BC);
for No:=1 to N do
    begin
    if MultiNodo then read(BC,VarNodo);
    if VarNodo.NumEjem>0 then
```

```
begin
 if MultiNodo then writeln
          ('Estoy en el nodo', No:3);
 with VarNodo  do
  begin
  v:=NumEnt;
  q:=NumSal;
  for p:=1 to 200*NumEjem do
    begin
    i:=random(NumEjem)+1;
    d:=0;
    c:=0;
    for j:=1 to v do
     d:=d+NodEjem[i].ValEnt[j]*
       NodRegla[j,NodEjem[i].ValSal];
    for k:=1 to q do
     begin
      d2:=0;
      if k<>NodEjem[i].ValSal then
      for j:=1 to v do
        d2:=d2+NodEjem[i].ValEnt[j]*
            NodRegla[j,k];
      if (d2>=d) and
         (k<>NodEjem[i].ValSal) then
          begin
           c:=c+1;
           for j:=1 to v do
             NodRegla[j,k]:=
               NodRegla[j,k]-
                 NodEjem[i].ValEnt[j]
          end;
       if c=1 then
          begin
           for j:=1 to v do
            NodRegla
              [j,NodEjem[i].ValSal]:=
              NodRegla
               [j,NodEjem[i].ValSal]+
                NodEjem[i].ValEnt[j];
            c:=c+1
           end
```

```
            end {for k}
          end   {for p}
        end;       {with}
        if MultiNodo then
            begin
              seek(BC,No-1);
              write(BC,VarNodo)
            end
      end {if NumEjem > 0}
    end   {for No}

END;

PROCEDURE conservar (No:Inodo; hj:Isal);

{Permite guardar en disco
 el ejemplo que se ha dado}

var j:Ient;
    respuesta:boolean;

BEGIN

write('Quieres guardar este ejemplo?   ');
pregunta(respuesta);
if respuesta then
  with VarNodo do
   begin
    NumEjem := NumEjem + 1;
    for j:=1 to NumEnt do
        begin
        if Entrada[j].Marca then
            begin
              write('Necesito completar');
              writeln(' la informacion:');
              pidevalor (j,Entrada[j].Valor)
            end;
        NodEjem[NumEjem].ValEnt[j] :=
                                Entrada[j].Valor
        end;
    NodEjem[NumEjem].ValSal := hj;
```

```
    if MultiNodo then
       begin
         seek(BC,No-1);
         write(BC,VarNodo)
       end
   end {if respuesta, with VarNodo}

END;

PROCEDURE corregir
     (No:Inodo; D: Decision; var hj:Isal);

{Actua en caso de que el sistema haya
 deducido un resultado incorrecto.
 Pregunta por el resultado correcto
 y por las variables que le faltan,
 y corrige las reglas}

var  hi,i   : Isal;
     j      : Ient;
     Num    : integer;

BEGIN

with VarNodo do
   begin
     for i:=1 to NumSal do
       writeln(i:3,'. ',Salida[i].Nombre);
     write('Cual es el numero del ');
     WRITE('resultado correcto?   ');
     leeentero(Num,NumSal);
     hi:=Num;
     for j:=1 to NumEnt do
         if Entrada[j].Marca then
           pidevalor (j,Entrada[j].Valor);
     for i:=1 to NumSal do
        if (D[i]>=D[hi]) and (i<>hi)
        then
         for j:=1 to NumEnt do
           NodRegla[j,i]:=NodRegla[j,i]-
                          Entrada[j].Valor;
```

```
      for j:=1 to NumEnt do
            NodRegla[j,hi]:=NodRegla[j,hi]+
                              .Entrada[j].Valor
   end; {with}
hj:=hi

END;
PROCEDURE examinar;

{Permite presentar en la pantalla los
ejemplos y las reglas de todos
los nodos}

var  pp          : Inodo;
     qq          : Ient;
     rr,
       SalBuena  : Isal;
     ss          : Iejem;
     ContLineas  : Integer;
     Respuesta   : boolean;

  procedure rotulo;

   begin

     ClrScr;
     write('Matriz de reglas, R[i,j]');
     writeln(', en el nodo' , pp:3);
     write('Con la columna j se calcula');
     write(' el valor de decision para');
     writeln(' el resultado j,');
     write('en funcion de los valores');
     writeln(' de las variables i');
     write('_____________________________');
     write('_____________________________');
     writeln('_________________')

   end;

BEGIN {examinar}
 if MultiNodo then reset(BC);
```

```
for pp:=1 to N do
 begin
  if MultiNodo then read(BC,VarNodo);
  with VarNodo do
   repeat
     ss:=0; respuesta:=true;
     while (ss<NumEjem) and respuesta do
      begin
       ss:=ss+1;
       limpiar;
       write('NODO ',pp:3);
       writeln(', EJEMPLO ',ss:3);
       writeln;
       for qq:=1 to NumEnt do
        begin
         write
           (qq:2,'.',Entrada[qq].Nombre,
            NodEjem[ss].ValEnt[qq]:9:3,
                                    ';  ');
          if (qq mod 2=0) or (qq=NumEnt)
             then writeln
        end;
       writeln;
       SalBuena:=NodEjem[ss].ValSal;
       write('Resultado: ');
       write(SalBuena:3,'.');
       writeln(Salida[SalBuena].Nombre);
       writeln('Resultados falsos:');
       for rr:=1 to NumSal do
       if (rr<>SalBuena) then
        begin
         write(rr:3,'.');
         write(Salida[rr].Nombre,' ':7);
         if (rr mod 2 = 0)
           or (rr = NumSal) then writeln
        end;
       writeln; write('Pulsa "S" para ');
       write(' seguir con el siguiente');
       writeln(' ejemplo y "N" para ');
       write(' pasar a ver las reglas');
       writeln(' de este nodo ');
```

```
          pregunta(respuesta)
         end; {while}
        rotulo;
        ContLineas:=0;
        for qq:=1 to NumEnt do
         begin
          if ContLineas >=20 then
           begin
            write('Cualquier tecla');
            write(' para seguir ');
            repeat until keypressed;
            rotulo;
            ContLineas:=0
           end;
          write('i=',qq:2,' ');
          for rr:=1 to NumSal do
           begin
            if (rr mod 8 = 1)and(rr>1) then
                                  write(' ':5);
            write(NodRegla[qq,rr]:9:3);
            if (rr mod 8 = 0)
                  or (rr = NumSal) then
              begin
               writeln;
               ContLineas:=ContLineas+1
              end
            end {for rr}
         end    {for qq};
        writeln; write('Pulsa n para ');
        write('volver a ver este nodo, y ');
        write('s para el siguiente  ');
        pregunta(respuesta);
      until respuesta
    end {for pp}
  END;

BEGIN
repeat
 menu(Opcion);
 case Opcion of
 1:inicia;
```

```
2:ejemplos;
3:ejercitar;
4:{entrenar}
  begin
   repeat
    write('Un momento, me inicializo');
    inicmarcas;
    if MultiNodo then reset(BC);
    limpiar;
    writeln('Sesion de entrenamiento. ');
    for ContNodo:=1 to N do
     begin
      if MultiNodo then read(BC,VarNodo);
      deducir(ContNodo,VectorDecision,
                        NumResult,NoDec);
      if NoDec then corregir(ContNodo,
                 VectorDecision,NumResult)
      else
       begin
       write('Es correcto?   ');
       pregunta(Resp);
       if not Resp then
        corregir(ContNodo,VectorDecision,
                              NumResult)
       end;
      conservar(ContNodo,NumResult);
      if (ContNodo < N) then
         actualizar(ContNodo,NumResult)
     end {for ContNodo};
    writeln;  write
      ('Quieres seguir entrenandome?  ');
    pregunta(Resp)
   until not Resp
  end;
5:{funcionamiento normal}
  repeat
   limpiar;
   write('Un momento, que ');
   writeln('me inicialice');
   inicmarcas;
   if MultiNodo then
```

```
      reset(BC);
    limpiar;
    write('Estoy preparado; te pedire ');
    write('informacion y tratare de ');
    writeln('deducir el resultado');
    for ContNodo:=1 to N do
     begin
      if MultiNodo then
                  read(BC,VarNodo);
      deducir(ContNodo,VectorDecision,
                        NumResult,NoDec);
      if not NoDec and (ContNodo < N)
       then
            actualizar(ContNodo,NumResult)
     end;
    writeln;
    write('Quieres seguir?     ');
    pregunta(Resp)
   until not Resp;

6:{designar fichero}
  begin
   limpiar;
   repeat
    write('Nombre del fichero? ');
    readln(F);
    buscafich(F,Resp);
    if not Resp then
       writeln('Ese fichero no existe');
   until Resp;
   if not MultiNodo then
          read(BC,VarNodo);
   N:=filesize(BC);
   MultiNodo:= N>1
  end;


7:examinar;

8:{cerrar ficheros y terminar}
  begin
```

```
    if not MultiNodo then
     begin
      reset(BC);
      write(BC,VarNodo)
     end;
    close(BC);
    write
      ('Quieres salir del programa?  ');
    pregunta(Resp)
   end

 end {case}

until Resp and (opcion = 8)

END.
```

El experto bayesiano

La que sigue es una versión del sistema descrito en el Capítulo 10. Se supone que la base de conocimientos (enfermedades y probabilidades) y el listado de evidencias (síntomas) se han introducido previamente en dos ficheros distintos, pero con el mismo formato con que figuran al final del Capítulo 10. (Obsérvese, a este respecto, que en dicho formato hemos puesto detrás de cada número, y antes de la coma que lo separa del siguiente, un espacio en blanco, que para el caso de utilizar BASIC es innecesario, pero que para la versión en Pascal nos permite hacer uso de la conversión automática de caracteres a binario con el procedimiento read(N).)

```
PROGRAM experto_bayesiano
                        (input,output,BC,BE);
CONST
     MaxEvid=99;          MaxHipot=99;
     MaxEvidHipot=1200;
       {99 Hipotesis maximo, con una
        media de unas 12 Evid/Hipot}
TYPE
     NumEvid=1..MaxEvid;
     NumHipot=1..MaxHipot;
     NumEvidHipot=1..MaxEvidHipot;

     RegEvid = record
               VR:real; {valor de la
                 evidencia segun el cambio
                 que puede provocar en las
                 probabilidades}
               ER:boolean   {true si ya se
                 ha preguntado por la evi-
                 dencia y false si no}
               end;
```

```
RegHipot= record
          Pinic,
           Pact:real; {probabilidades
               a priori inicial y
               actualizada}
          MA,MI:real; {valores maximo
               y minimo que puede
               llegar a tomar Pact en
               en funcion de las evi-
               dencias aun no conoci-
               das}
          EN,EP:NumHipot {numeros de
               evidencias aplicables
               a la hipotesis, total
               y pendientes}
          end;
RegProb = record
            i     : NumHipot;
            j     : NumEvid;
            Ps,Pn: real
                 {P(j:i) y P(j:no i)}
          end;

VAR
    Opcion,
       ContLin    :integer;
    Resp,Error    :boolean;
    NEvid         :NumEvid;
    Hmax,
      NHipot,iii  :NumHipot;
    NEvidHipot,jjj:NumEvidHipot;
    Evidencia     :array[1..MaxEvid]
                              of RegEvid;
    Hipotesis     :array[1..MaxHipot]
                             of RegHipot;
    Probabilidades:array[1..MaxEvidHipot]
                             of RegProb;
    BC,BE         :text;
```

```
PROCEDURE pregunta(var SiNo:boolean);

var car:char;

BEGIN
repeat
 readln(car);
 if car in ['s','S']
    then SiNo:=true
    else
     if car in ['n','N']
      then SiNo:=false
      else
       write('Tienes que pulsar S o N   ')
until car in ['s','n','S','N']
END;

PROCEDURE cabecera;

BEGIN
 ClrScr;
 writeln
     ('SOY EL SISTEMA EXPERTO BAYESIANO');
   writeln
     ('_____________________________________');
   writeln
END;

PROCEDURE AsignaFich(caso:boolean);

{Asigna un fichero para la base de cono-
 cimientos (caso=false) o para la lista
 de evidencias (caso=true)}

var
   Nombre:string[14];
   existe:boolean;

BEGIN
  write('Nombre del fichero?   ');
  repeat
```

```
    readln(Nombre);
    if caso then begin
                    assign(BE,Nombre);
                    {$I-} reset(BE) {$I+}
                    end
            else begin
                    assign(BC,Nombre);
                    {$I-} reset(BC) {$I+}
                   end;
    existe:=(IOResult=0);
    if not existe then write
      ('Fichero inexistente .Otro nombre ')
  until existe
END;

PROCEDURE menu (var op:Integer);

BEGIN
  cabecera;
  write('1. Designar el fichero de la ');
  writeln('base de conocimientos');
  write('2. Designar el fichero de la ');
  writeln('lista de evidencias');
  writeln('3. Ejecutar el experto');
  write('4. Examinar valores iniciales ');
  write('y actuales de las probabili');
  writeln('dades de las hipotesis');
  write('5. Examinar las probabilidades');
  write(' condicionales de la base ');
  writeln('de conocimientos');
  writeln('6. Salir del programa');
  writeln;write('Escoge un numero   ');
  repeat
    readln(op);
    if (op<1) or (op>6) then write
     ('Solo acepto un numero entre 1 y 6 ')
  until (op>0) and (op<7)
END;
PROCEDURE CalculNevid;

{Lee el fichero de la lista de
```

```
 evidencias, y las cuenta}

var linea:string[80];

BEGIN
  Nevid:=0;
  repeat
    readln(BE,linea);
    if (linea[1]>='0') and (linea[1]<='9')
       then  Nevid:=Nevid+1
  until eof(BE)
END;

PROCEDURE CargaProb;

{Carga en la memoria las probabilidades
 contenidas en la base de conocimientos}

var
      IHipot      : NumHipot;
      IEvidHipot  : NumEvidHipot;
      NError      : integer;
      aborto      : boolean;

   procedure saltacoma;
   {Lee caracteres hasta encontrar una
    coma}

   const coma=',';
   var car: char;

   begin
       repeat
          read(BC,car)
       until (car=coma) or eof(BC)
   end;

   procedure CargaHipot
                  (var desborda:boolean;
                       IHipot:NumHipot);
   {Carga las probabilidades referentes a
```

```
    la hipotesis IHipot)}

   var
      fin          : boolean;
      ContEvid,
      IEvid        : NumEvid;
      P0,Psi,Pno   : real;
      Numero       : integer;
begin  {CargaHipot}
{$I-} read(BC,P0); {$I+}
if IOResult<>0 then
   begin
    NError:=NError+1;
    writeln('Error de formato en la ',
            'hipotesis num.',IHipot:3);
    writeln('Pongo P0=0');
    P0:=0
   end;
Hipotesis[IHipot].Pinic:=P0;
fin:=false;
ContEvid:=0;
repeat
 saltacoma;
 {$I-} read(BC,Numero); {$I+}
 if IOResult<>0 then
  begin
   NError:=NError+1;
   writeln('Error de formato en la ',
           'hipotesis num.',IHipot:3);
   writeln('Ignoro la evidencia que ',
           'hace el num.',ContEvid+1:3,
           ' de esta hipotesis');
   saltacoma; saltacoma
  end
 else if Numero=999
   then fin:=true
   else if IEvidHipot=MaxEvidHipot
    then
     begin
      writeln('Se desborda la ',
              ' capacidad de memoria',
```

```
                'reservada');
      fin:=true;
      desborda:=true
     end
     else
      begin
       IEvid:=Numero;
       ContEvid:=ContEvid+1;
       saltacoma;
       {$I-}read(BC,Psi);{$I+}
       if IOResult<>0 then
        begin
         NError:=NError+1;
         writeln('Error de formato en ',
                 'la hipotesis',IHipot:3,
                 ', Evidencia',IEvid:3);
         writeln('Pongo Psi=0.5');
         Psi:=0.5
        end;
       saltacoma;
       {$I-}read(BC,Pno);{$I+}
       if IOResult<>0 then
        begin
         NError:=NError+1;
         writeln('Error de formato en ',
                 'la hipotesis',IHipot:3,
                 ', Evidencia',IEvid:3);
         writeln('Pongo Pno=0.5');
         Pno:=0.5
        end;
       IEvidHipot:=IEvidHipot+1;
       with Probabilidades[IEvidHipot] do
         begin
           i:=IHipot;
           j:=IEvid;
           Ps:=Psi;
           Pn:=Pno
         end
     end  {IevidHipot<MaxEvidHipot}
until fin;
Hipotesis[IHipot].EN:=ContEvid
```

```
  end;    {CargaHipot}

BEGIN   {CargaProb}

  IHipot:=0;
  IEvidHipot:=0;
  NError:=0;
  aborto:=false;
  reset(BC);
  saltacoma;
  while not eof(BC)
        and not aborto
        and (NError<50) do

    begin
      IHipot:=IHipot+1;
      CargaHipot(aborto,IHipot);
      saltacoma
    end;

  if NError>0 then writeln
       ('detectados',NError:3,' errores');
  NHipot:=IHipot;
  NEvidHipot:=IEvidHipot

END;

PROCEDURE inicia;

{Inicializa las estructuras de datos
 y calcula los valores iniciales del
 vector VR}

var
   ContHipot     : NumHipot;
   ContEvid      : NumEvid;
   ii            : NumEvidHipot;
   P             : real;

BEGIN
  for ContEvid:=1 to Nevid do
```

```
    with Evidencia[ContEvid] do
      begin
       VR:=0;  ER:=false
      end;
      ii:=1;
  for ContHipot:=1 to NHipot do
      begin
       with Hipotesis[ContHipot] do
         begin
           Pact:=Pinic; P:=Pact; EP:=EN
         end;
       while
          (Probabilidades[ii].i=ContHipot)
           and (ii<=NEvidHipot) do
          with Probabilidades[ii] do
            begin
               Evidencia[j].VR:=
                 Evidencia[j].VR+abs(P*Ps/
                 (P*Ps+(1-P)*Pn)-P*(1-Ps)/
                 (P*(1-Ps)+(1-P)*(1-Pn)));
               ii:=ii+1
            end
      end  {for ContHipot}

END;
PROCEDURE deduce
               (var HipotMaxMin:NumHipot);

{Pregunta por evidencias y deduce
 la hipotesis mas probable}

var
    RE              : real;
    acaba,
    equivocado      : boolean;
    ContEvid,
    Emax            : NumEvid;
    ContHipot       : NumHipot;
    ContEvidHipot   : NumEvidHipot;

procedure CalculMax(var Max:NumEvid);
```

```
{Busca la evidencia mas importante}

var R:real;

begin

  R:=0;
  Max:=0;
  for ContEvid:=1 to Nevid do
    with Evidencia[ContEvid] do
      begin
        if VR>R then
          begin
           R:=VR;
           Max:=ContEvid
           end;
        VR:=0
      end

end  {CalculMax};
procedure PregEv
         (Ev:NumEvid; var respuesta:real);
   {Pregunta por la evidencia de numero Ev
    y recoge la respuesta}
   var     linea        : string[80];
           NumAlfa      : string[2];
           NumNum, Cod  : integer;
           fin          : boolean;
   begin
     reset(BE);
     fin:=false;
     repeat
       if eof(BE)
        then
         begin
          writeln('Inconsistencia en la ',
                 'base de conocimientos:',
                 ' el sintoma numero',
                 Ev:3,' no existe');
          fin:=true;
          equivocado:=true
```

```
        end
      else
        begin
         readln(BE,linea);
         if (linea[1]>='0') and
            (linea[1]<='9') then
           begin
            NumAlfa:=copy(linea,2,2);
            val(NumAlfa,NumNum,Cod);
            if (Cod<>0) then
             begin
              writeln('Error de formato',
               ' en la lista de eviden',
               'cias(num. de evidencia:',
               NumAlfa,')');
              fin:=true;
              equivocado:=true
             end
          else {Cod = 0}
             if NumNum=Ev then
              begin
               delete(linea,1,13);
               writeln(linea);
               readln(BE,linea);
               while (linea[1]=' ')
                     and not eof(BE) do
                begin
                 writeln(linea);
                 readln(BE,linea)
                end;
               fin:=true;
               Evidencia[Ev].ER:=true
              end
         end  {if linea}
      end    {no eof}

until fin;

if equivocado then
  writeln('No sigo')
  else
```

```
      repeat
       readln(respuesta);
       if (respuesta >5) or
          (respuesta<-5) then
         write
         ('Dar valor entre -5 y +5    ')
      until (respuesta<=5)
        and (respuesta>=-5)
end  {PregEv};
procedure ActualizaP;

   {Obtiene nuevos valores para las
    probabilidades de las hipotesis,
    teniendo en cuenta la respuesta
    dada a la evidencia Emax}

   var P,P1,P2: real;

   begin

    for ContEvidHipot:=1 to NEvidHipot do

     with Probabilidades[ContEvidHipot] do

     if (j=Emax) and (Hipotesis[i].EP>0)
      then

       begin
        Hipotesis[i].EP:=
                      Hipotesis[i].EP-1;
        P:=Hipotesis[i].Pact;
        P1:=P*Ps + (1-P)*Pn;
        P2:=(Ps-P1)*RE/5;
        if RE>0 then
                 P:=P*(1+P2/P1);
        if RE<0 then
                 P:=P*(1+P2/(1-P1));
        if(P=0) or (P=1) then
                 Hipotesis[i].EP:=0;
        Hipotesis[i].Pact:=P
       end
```

```
end  {ActualizaP};
procedure CalculVR;
   {Calcula nuevos valores de reglas y
    valores maximos y minimos de las
    probabilidades posibles en funcion
    de las evidencias aun desconocidas}

   var MaxMin,A1,A2,A3,
           A4,P,Psi,Pno  : real;
       ii                :  NumEvidHipot;

   begin
    acaba:=true;
    MaxMin:=0;
    HipotMaxMin:=0;
    ii:=1;
    for ContHipot:=1 to NHipot do
     begin
     A1:=1;  A2:=1;  A3:=1;  A4:=1;
     P:=Hipotesis[ContHipot].Pact;
     if Hipotesis[ContHipot].EP>0 then
        while
          (Probabilidades[ii].i=ContHipot)
           and (ii<=NEvidHipot) do
           begin
            with Probabilidades[ii] do
            if not Evidencia[j].ER then
             begin
              if Pn>Ps then
                begin
                 Psi:=1-Ps;
                 Pno:=1-Pn
                end
              else
                begin
                 Psi:=Ps;
                 Pno:=Pn
                end;

             Evidencia[j].VR:=
             Evidencia[j].VR+P*Psi/
```

```
            (P*Psi+(1-P)*Pno)-P*(1-Psi)/
            (P*(1-Psi)+(1-P)*(1-Pno));
            A1:=A1*Psi;
            A2:=A2*Pno;
            A3:=A3*(1-Psi);
            A4:=A4*(1-Pno)
           end {if not Evidencia[j].ER};
          ii:=ii+1
         end      {while}
   else
     repeat
      ii:=ii+1
     until
        Probabilidades[ii].i<>ContHipot;
   with Hipotesis[ContHipot] do
    begin
     MA:=Pact*A1/(Pact*A1+(1-Pact)*A2);
     MI:=Pact*A3/(Pact*A3+(1-Pact)*A4);
     if MA<Pinic then EP:=0;
     if MI>MaxMin then
      begin
       HipotMaxMin:=ContHipot;
       MaxMin:=MI
      end
    end  {with Hipotesis}
   end;   {for ContHipot}
   for ContHipot:=1 to NHipot do
    if ContHipot<>HipotMaxMin then
      with Hipotesis[ContHipot] do
       if (MaxMin<=MA)
        then acaba:=false
        else EP:=0
end;

BEGIN   {deduce}

  equivocado:=false;
  cabecêra;
  writeln('Responde a las preguntas con ',
          'un numero comprendido entre ',
          '-5(no) y 5(si)');
```

```
  writeln;
  repeat

    CalculMax(Emax);
    if Emax>0
     then

      begin
       PregEv(Emax,RE);
       ActualizaP;
       CalculVR
       end

     else

      begin
       writeln
        ('No tengo bastantes evidencias');
       equivocado:=true
      end

  until acaba  or equivocado

END;

PROCEDURE resultado( H:NumHipot);
{Lee la base de conocimientos en disco
 y escribe el nombre correspondiente a
 la hipotesis numero H}

var   NumNum,Cod : integer;    ii : 1..80;
      linea      : string[80];
      NumAlfa    : string[2];
      fin        : boolean;
BEGIN
 reset(BC); writeln;writeln;writeln;
 fin:=false;
 repeat
  readln(BC,linea);
  if (linea[1]>='0') and (linea[1]<='9')
   then
```

```
    begin
     NumAlfa:=copy(linea,2,2);
     val(NumAlfa,NumNum,Cod);
     if NumNum=H then
      begin
       write('La hipotesis mas ',
             'probable es: ');
       fin:=true; ii:=11;
       while (linea[ii]<>',') do
        begin
         write(linea[ii]:1); ii:=ii+1
        end;
       writeln(',');
       writeln('con una probabilidad = ',
              Hipotesis[H].Pact:10:8)
      end
    end
 until eof(BC) or fin;
 if eof(BC) and not fin then
   writeln('Inconsistencia en la base de',
       ' conocimientos: la hipotesis de ',
      'numero ',H:3,' no existe')
END {resultado};

BEGIN    {experto_bayesiano}
  repeat
    menu(Opcion);
    case Opcion of

    1:
      begin
        AsignaFich(false);
        writeln('Espera un momento: ',
        'voy a cargar la base de ',
        'conocimientos');
        CargaProb
      end;

    2:
      begin
        AsignaFich(true);
```

```
      writeln('Espera un momento: ',
      'voy a contar el numero de ',
      'evidencias');
      CalculNevid
     end;

  3:
    begin
      {$I-} reset(BC); {$I+}
      Error:=IOResult<>0;
      if Error then
         begin
          writeln('Tienes que asignar ',
          'previamente el fichero',
          'de la base de ',
          'conocimientos');
          Error:=true
         end;
      {$I-} reset(BE); {$I+}
      if IOResult<>0 then
          if Error
           then

             begin
              writeln('Y tambien el ',
              'de los nombres de ',
              ' evidencias')
             end

           else
             begin
              writeln('Tienes que ',
               ' asignar previamente',
               ' el fichero de los ',
               'nombres de evidencias');
              Error:=true
             end;

      if not Error then
        repeat
         inicia;
```

```
        deduce(Hmax);
        resultado(Hmax);
        write('Quieres que empiece ',
        'otra vez? ("Que pase',
        ' el siguiente!")     ');
        pregunta(Resp)
       until not Resp

    end;

4: begin
   ContLin:=0;
   ClrScr;
   writeln('HIPOTESIS','Pinic':15,
           'Pact':19,'EN':15,'EP':7);
   for iii:=1 to NHipot do
     begin
     if ContLin>=20 then
       begin
       writeln('cualquier tecla');
       repeat until keypressed;   ClrScr;
       writeln('HIPOTESIS','Pinic':15,
            'Pact':19,'EN':15,'EP':7);
       ContLin:=0
       end;
     with Hipotesis[iii] do
     writeln(iii:6,' ':5,Pinic:15:10,
         ' ':5,Pact:15:10,EN:11,EP:7);
     ContLin:=ContLin+1
     end;
     writeln('cualquier tecla');
     repeat until keypressed
   end;
5: begin
   ClrScr;
   writeln('HIPOTESIS  EVIDENCIA',
    ' ':8,'P(E:H)',' ':14,'P(E:noH)');
   ContLin:=0;
   for jjj:=1 to NEvidHipot do
     begin
     if ContLin>=20 then
```

```
      begin
       writeln('cualquier tecla');
       repeat until keypressed;   ClrScr;
       writeln('HIPOTESIS  EVIDENCIA',
           'P(E:H)':16,'P(E:noH)':18);
       ContLin:=0;
      end;
          with Probabilidades[jjj] do
          writeln(i:6,j:11,' ':5,Ps:15:10,
                              ' ':5,Pn:15:10);
          ContLin:=ContLin+1
          end;
          writeln;writeln;
          writeln('Card(HipotxEvid)=',
                       NevidHipot:3);
          writeln('cualquier tecla');
          repeat until keypressed
        end;

    6: begin
       writeln;
       writeln
           ('Adios, ha sido un placer');
       close(BC);
       close(BE)
       end

    end  {case}
  until Opcion=6
END.
```

OTROS TITULOS DE LA EDITORIAL DIAZ DE SANTOS, S.A.

ZX SPECTRUM. COMO OBTENER EL MAXIMO RENDIMIENTO.
Sinclair, Ian

SPECTRUM. LIBRO DE JUEGOS.
James, M; Gee, S.M., y Ewbank, K.

EL PROGRAMADOR DE SPECTRUM.
Gee, S.M.

CUARENTA JUEGOS EDUCATIVOS PARA EL SPECTRUM.
Apps, Vince

SPECTRUM. INTRODUCCION AL CODIGO MAQUINA.
Cómo obtener más velocidad y potencia.
Sinclair, Ian

OBTENGA EL MAXIMO RENDIMIENTO DE SU ZX MICRODRIVE.
Sinclair, Ian

EL PROGRAMADOR DE DRAGON.
Gee, S.M.

INTRODUCCION A LAS REDES LOCALES DE INFORMATICA APLICADA.
Gee, K.C.E.

INFORMATICA JURIDICA DOCUMENTAL.
López-Muñiz, Miguel

COMPILADORES E INTERPRETES.
Un enfoque pragmático.
Valverde Andreu, A. y Sánchez Dueñas, Gonzalo

LOS ORDENADORES. FUNDAMENTOS Y SISTEMAS.
Giarratano, Joseph

CONCEPTOS ACTUALES SOBRE LA TECNOLOGIA DE LOS ORDENADORES.
Giarratano, Joseph

INTRODUCCION A LAS BASES DE DATOS RELACIONALES.
Mayne y Wood

INTRODUCCION AL LOGO.
Alan, Boris

LOS ORDENADORES DE LA QUINTA GENERACION.
Simons, G.L.

UNIX.
Guía profesional.
Banaham y Rutter

DICCIONARIO OXFORD DE INFORMATICA.
Oxford University

LENGUAJE C.
Guía de programación
Purdum

SISTEMAS EXPERTOS.
Conceptos y ejemplos
Alty, J.L. y Coombs, M.J.

COMMODORE 64.
Juegos Educativos
Noble, T.

SPECTRUM.
Diseño de juegos
Ewbank, James y Gee

EN PREPARACION

PRINCIPIOS DE INTELIGENCIA ARTIFICIAL.
Nilsson

TRABAJE CON LA d. BASE II
M. de Pace

Un sistema experto no es otra cosa que un programa de ordenador dotado de la experiencia humana en algún campo de conocimiento. Hay sistemas expertos que ayudan a los médicos a diagnosticar y tratar enfermedades, otros que pueden deducir estructuras químicas, otros que determinan el lugar más indicado para perforar en busca de minerales. Hasta ahora, todo esto estaba reservado a los grandes ordenadores, a los programas muy complejos y a los lenguajes exóticos. Pero este libro, con explicaciones sencillas y claras y con programas en BASIC, permite que cualquier usuario de ordenadores personales pueda abordar el tema.

El lector se introducirá así en el debate, tan actual, sobre la Inteligencia Artificial. Verá claramente cómo funciona un sistema experto y podrá utilizar su ordenador personal para que le ayude en tareas tales como la predicción meteorológica, la reparación de automóviles o, incluso, el diagnóstico médico. Todos los programas incluidos en el libro se han comprobado sobre los conocidos micros Apple II y Sinclair Spectrum.

He aquí algunos de los comentarios publicados en revistas prestigiosas:

«¿Puede pedirse más de lo que se encuentra en las páginas de **Construya su propio sistema experto** de Chris Naylor?»

Computing Today

«El libro de Naylor constituye una buena introducción al tema, e incluye descripciones de la mayoría de los sistemas expertos conocidos que actualmente se utilizan.»

The Times

«Es un libro excelente sobre un tema formidable. Aun en el caso de que Vd. no llegue a construir un sistema experto, no habrá malgastado su dinero.»

Personal Computer News

«Probablemente, éste será el libro de texto más insólito que jamás haya leído. Si usted ya está aburrido y buscando algo nuevo, no podría encontrar nada mejor.»

Which Micro & Software Review

«Construya su propio sistema experto, de Chris Naylor, es uno de los libros más interesantes que he leído últimamente.»

Popular Computing Weekly

DIAZ DE SANTOS, S.A.

MADRID / BARCELONA